

# Benchmark资讯自动播报Agent调研报告

## 1. 自动发现相关Benchmark的工具和项目

**论文/论文库监控：**利用论文API和开源工具可以自动发现新发布的研究成果。例如，arXiv提供官方API和RSS订阅，支持按关键词或分类检索最新论文<sup>1</sup>。很多开源项目实现了对arXiv的新论文监控并推送的功能，例如 **arxiv-slack-bot**：这是一个Python工具，可以检索arXiv新论文信息并发送摘要到Slack<sup>2</sup>。类似地，Stanford VL的**ArxivBot**、discus0434的**paper-summarizer**等项目也通过订阅arXiv每日更新、结合LLM生成摘要后在Slack推送<sup>3</sup>。这些工具通常支持自定义过滤关键词、作者等，从庞杂的新论文中筛选出可能引入新Benchmark的数据集或任务。

**评测榜单/基准库监控：**Papers with Code (PwC) 提供了公开API客户端和数据集，可用于追踪新添加的任务和数据集及其SOTA结果<sup>4</sup>。通过PwC的API，我们可以获取最新的benchmark任务信息、关联论文和代码库等。除了PwC，近期出现的AgentBench、HELM等综合性评测平台也值得监控：**AgentBench**是专为LLM-Agent评测设计的多场景基准集合<sup>5</sup>；**HELM** (Holistic Evaluation of Language Models) 由斯坦福推出，开放源代码框架，提供一系列场景和指标的持续评测，被称为“活的Benchmark”<sup>6</sup>。通过调用这些平台的开放数据接口或抓取其更新日志，可以及时发现新的Benchmark场景或榜单变化。尤其HELM作为开源框架，可以获取到不同模型在各种场景下的最新表现数据<sup>7</sup>。

**开源社区趋势监控：**GitHub和Hugging Face等社区是Benchmark工具和数据集发布的重要来源。可以利用GitHub的搜索或趋势接口发现热门的新项目，例如按关键词“benchmark”搜索近期创建的项目，或通过非官方的Trending API获取每日趋势仓库<sup>8</sup>。还有一些工具可跟踪仓库Stars增长历史，用于判断项目热度<sup>9</sup>。Hugging Face Hub提供Python客户端 `huggingface_hub`，支持按标签或更新时间列出新发布的数据集/模型，可用来发现**新Benchmark数据集**（如带有“benchmark”标签的数据集）。此外，定期扫描ACL Anthology、OpenReview等学术源以获取最新**会议Benchmark**（ACL、NeurIPS等可能有新竞赛任务）也是可行的。

**社交媒体和内部线索：**对于Twitter (X) 等社交媒体上研究者发布的新Benchmark信息，可使用Twitter API或开源爬取工具（如sncrape）按特定账号或关键词跟踪。例如，有项目每日汇总特定Twitter帐号提及的论文并在Slack推送<sup>10</sup>。内部团队线索方面，可利用Slack API监控团队讨论频道中的关键词触发——比如当有人分享新的Benchmark链接时，Agent自动记录下来。此外，RSS聚合也是简便方案：Slack本身内置RSS应用，可以订阅如arXiv、GitHub Release等RSS源实现即时播报<sup>11</sup><sup>12</sup>。综上，多源头的自动发现需要组合使用论文API、榜单接口、社区爬虫等工具来确保不遗漏新出现的Benchmark资讯。

## 2. Benchmark预筛选模块

自动发现的候选Benchmark往往数量众多且质量不一，需要一个**预筛选模块**对其进行结构化评分，筛选出高价值的候选。可考虑以下维度及相应工具支持：

- **活跃度指标：**利用GitHub API获取代码仓库的star增长和最近commit时间，以衡量项目热度和维护情况（例如过去一周stars增速）<sup>13</sup>。也可查询GitHub Trending数据<sup>8</sup>或最近发布版本活动。对Hugging Face数据集，可查看下载量趋势等作为活跃度参考。工具方面，可以使用Python的 `PyGithub` 库或GitHub GraphQL接口批量查询stars、forks和最近更新日期。
- **可复现性：**检查Benchmark是否提供了开源代码、数据或运行脚本。PwC的数据通常会标注论文是否附带代码，以及提供的代码仓库链接。可以编写规则扫描论文摘要或项目README中是否有“我们发布了代码在GitHub”“提供了数据集下载”等字样；也可以借助LLM从描述中提取是否提及开放资源（类似

**LLM辅助信息抽取**)。例如,使用LangChain的结构化提取链,提示GPT从论文简介提取「是否提供代码或数据」字段。另外,像**AgentBench**、**HELM**本身强调可复现评测,若是这些来源的新任务,通常可认为具备可复现性<sup>6</sup>。

- **许可合规**: 通过仓库API获取项目的开源许可证(GitHub API会返回license字段)。也可利用SPDX库检查license是否为允许内部使用的类别。对于数据集,可查看Hugging Face等提供的许可证标记。预筛模块应自动过滤掉许可证不合适的候选。例如如果license为GPL类而内部不便使用,则降低评分。
- **任务新颖性**: 判断Benchmark任务类型在现有体系中是否新颖。可将新Benchmark的任务描述与内部已有任务库比对,衡量相似度。具体实现上,可以构建任务描述Embedding库,用向量相似度判断新任务是否与已有任务高度重复。另一路径是用LLM分析:让LLM阅读Benchmark简介,提取任务类型,然后询问“该任务是否提出了新的问题或场景”。例如“这是一个全新的GUI自动化任务”则认为新颖度高。开源项目**Infogent**提出的三模块架构(Navigator-Extractor-Aggregator)也表明,利用LLM Extractor模块可从网页内容中抓取关键信息供决策<sup>14</sup>——在此情境下,Extractor可以帮助识别任务是否创新。
- **内部适配度(MGX场景相关)**: 这是内部自定义维度,需要根据公司产品或研究方向评估。可以预先定义一些规则或利用LLM判断。例如,如果内部代号“MGX”关注人机对话,那么在Benchmark描述中寻找对话/多智能体协作等关键词。LLM可以被提示:“阅读这段Benchmark描述,判断其与我们的‘MGX’项目的相关性高不高”,返回一个评分。

预筛模块可以综合以上维度形成每个候选的评分卡。许多步骤可以借助LLM**结构化抽取**来完成:例如用GPT总结项目的任务类别、提取仓库链接和发布日期等<sup>15</sup>。已有一些Agent式框架支持类似抽取流程,如**LangChain**可以结合自定义Prompt和输出Parser,将非结构化文本解析为JSON格式字段。最终,根据评分设定阈值,筛选低活跃度、不可复现或无关的条目,只保留“活跃且有代码”、“任务新颖且相关”的Benchmark进入后续流程。

### 3. 自动入库与播报流程

筛选合格的Benchmark候选后,需要将信息统一格式保存,并实现通知与人工确认机制。以下是推荐的实现方案:

- **统一格式化入库**: 将每条Benchmark候选记录存入一个结构化的数据存储,如Airtable表、Notion数据库或CSV文件等。Notion和Airtable均提供便捷的API和SDK,例如Notion API通过创建页面可以插入结构化字段。一个实战案例是开源项目**ArXivNotificator**,其使用Notion的数据库来记录推荐的论文列表<sup>16</sup>。我们可以借鉴其做法,为Benchmark建立类似的Notion表,字段包括名称、来源链接、概要、评分以及审核状态等。在实现上,使用官方库(如 `notion-client` 或 `pyairtable`)根据筛选结果自动插入记录<sup>17</sup>。对于CSV或内部数据库,也可以用Python直接写入。**统一格式**便于后续管理和查询,例如在Notion上可以方便地筛选和手动编辑。
- **Slack/邮件播报**: 集成Slack机器人的消息推送功能,将新增的候选Benchmark及时通知相关团队。在Slack上,可以通过Webhook直接发送消息,或使用Slack SDK以富文本格式发送卡片消息。其中包含Benchmark名称、简介、评分摘要,以及“添加”按钮。一些开源Slack机器人示例证明了可行性:如前述的arxiv-slack-bot会将摘要发送到Slack<sup>2</sup>,并支持进一步交互讨论<sup>18</sup>。我们可参考其实现,发送消息时附带交互按钮。当用户点击“一键添加”按钮时,Slack会调用预设的交互回调URL,我们的Agent后端收到事件后,可将该Benchmark的状态从“候选”更新为“已确认”,或直接移入正式的Benchmark清单。这实现了**人机协作的快速决策**。对于邮件通知,则可用SMTP或第三方服务发送定期汇总邮件,包含新候选Benchmark列表及链接,供决策者线下查看。

- **一键添加机制**：正如上文所述，通过Slack消息的互动组件（Interactive Message）可以实现一键操作。例如Slack的**Actions**允许在消息中放置button，点击后触发一个HTTP请求到自定义服务。在该服务中，我们编写逻辑：根据消息携带的Benchmark标识，在结构化表中将其标记为正式收录。Notion的场景下，可将记录的“状态”字段改为“已添加”，或将记录移动到正式数据库。Airtable类似可以修改记录或复制记录到另一表。整个过程响应迅速，用户无需离开Slack。开源项目中已有类似理念的实现，例如有人开发Slack机器人，可以点按钮将推荐的论文加入个人阅读清单（集成Notion）<sup>19</sup>。这些机制保证了**人工反馈融入自动流程**：模型负责发现和初筛，人负责最终确认，避免无关信息入库。

## 4. 调度与 workflow 组合方案

为了持续稳定地运行上述流程，我们需要调度和编排工具。推荐采用**定期调度 + 工作流编排**相结合的方案：

- **定时调度**：可以使用 **GitHub Actions** 的 `schedule` 触发，设定每天或每周定时运行爬取+筛选+播报流程。GitHub Actions 易于与代码仓库集成，配置YAML即声明工作流定时触发，适合轻量级的定期任务。另外一种选择是企业内部使用**Apache Airflow**或**Apache DolphinScheduler**等调度系统，编写DAG定时执行各Task（如“抓取论文->分析筛选->写入表->发送通知”链条）。Airflow具备可视化监控和重试机制，适合复杂任务的管理。如果需要近实时的监控（streaming运行），也可以考虑**消息队列触发**：例如检测到GitHub上特定事件（新repo或release）时，通过Webhook通知我们的Agent立即处理。
- **工作流框架**：在实现细节上，可利用一些Agent或管道框架来串联步骤。例如 **LangChain** 可以管理调用外部API、调用LLM分析、再调用数据库的序列，使代码更语义化。也可以选择**Prefect**或**Dagster**这类Python工作流库，方便地将数据抓取、LLM解析、入库、通知各步骤封装为任务，并处理依赖和错误。无论哪种实现，关键是模块清晰：数据采集模块、预筛选模块、通知模块等解耦，便于维护和扩展。

**推荐方案优势**：综上，我们建议组合使用“**API抓取 + LLM抽取 + 自动化运维**”的全流程方案：利用官方API和开源爬虫获取多源数据，借助LLM和规则进行智能筛选，然后通过结构化数据库和Slack完成播报与人工确认。这样的方案结合了自动发现的广度和智能筛选的精度。在实践中已有类似系统验证了各环节的有效性，例如前述ArXivNotificator同时用到了arXiv API、OpenAI的API进行内容筛选、Notion记录和Slack推送<sup>15</sup>。我们的方案通过模块化设计（类似Infogent提出的Navigator-Extractor-Aggregator架构<sup>14</sup>），保证每步可独立改进。定期调度和流水线框架则确保系统可靠运行、可监控和扩展。总体而言，该组合方案能够**高效覆盖最新Benchmark信息**并自动呈现给团队，同时保留人工决策把关，一键操作提升工作效率。各组件多采用成熟的开源工具和API，实现起来成本低且具备社区支持，是构建Benchmark资讯自动播报Agent的理想选择。

**参考来源：**

- Papers with Code官方API客户端<sup>4</sup>
- arXiv Slack机器人示例<sup>2 3</sup>
- Infogent信息聚合Agent框架<sup>14</sup>
- ArXivNotificator集成Slack/Notion/LLM案例<sup>15 16</sup>
- Stanford HELM评测框架（开源）<sup>6</sup>
- GitHub Trending API项目<sup>8</sup>

- 1 arXiv API Basics  
<https://info.arxiv.org/help/api/basics.html>
- 2 18 GitHub - t0d4/arxiv-slack-bot: Retrieves information of papers posted on arxiv.org and send its summary to Slack.  
<https://github.com/t0d4/arxiv-slack-bot>
- 3 10 GitHub - discuss0434/paper-summarizer: A Slack Bot for summarizing arXiv papers, powered by OpenAI LLMs.  
<https://github.com/discuss0434/paper-summarizer>
- 4 GitHub - paperswithcode/paperswithcode-client: API Client for paperswithcode.com  
<https://github.com/paperswithcode/paperswithcode-client>
- 5 AgentBench: Benchmarking Autonomous LLM Agents  
<https://www.emergentmind.com/topics/agentbench>
- 6 GitHub - stanford-crfm/helm: Holistic Evaluation of Language Models (HELM) is an open source Python framework created by the Center for Research on Foundation Models (CRFM) at Stanford for holistic, reproducible and transparent evaluation of foundation models, including large language models (LLMs) and multimodal models.  
<https://github.com/stanford-crfm/helm>
- 7 Holistic Evaluation of Language Models (HELM) - Stanford CRFM  
<https://crfm.stanford.edu/helm/latest/>
- 8 GitHub - NiklasTiede/Github-Trending-API: This API provides Data about Trending Repositories and Developers on Github.  
<https://github.com/NiklasTiede/Github-Trending-API>
- 9 emanuelef/daily-stars-explorer - GitHub  
<https://github.com/emanuelef/daily-stars-explorer>
- 11 12 Free personalized feed aggregator using Slack  
<https://blog.ristic.in.rs/2024/03/creating-personalized-feed-aggregator.html>
- 13 Unmasking the GitHub Star History: Track Daily Trends ... - Medium  
<https://medium.com/@emafuma/how-to-get-full-history-of-github-stars-f03cc93183a7>
- 14 Infogent: An Agent-Based Framework for Web Information Aggregation  
<https://arxiv.org/html/2410.19054v1>
- 15 16 17 19 arXivで公開された論文をslackから推薦するシステムを作ってみた  
<https://zenn.dev/iyo/articles/dd1151d3c9d129>