实名揭发 19 级网络空间安全专业杨淯而同学破坏计算机技术氛围、学术不端等的行为

本文编写组

July 18, 2022

目录

1 前言

2 杨淯而对计算机现有体系的无知

2.1 著作等身

“民科”一词是“民间科学家”的简称，与“官科”相对，原意是指未经历过正规的教育，通过自己的学习获得了知识并有相关研究成果的人的统称，是一个中性的词语。在过往，不乏有“民科”依靠自己的努力得出研究成果并得到学界公认者。

但在当下的网络语境中，“民科”一词已经慢慢演变为贬义词。近年来，不断有“民科”以各种包括名声与钱在内的各种动机发表耸人听闻的“证明”、“理论”、“见解”、“学说”、“反驳”、“推翻”等，典型的言论有“推翻相对论”“数学大厦轰然倒塌”等，然而绝大部分“民科”所发表的理论不外乎符合下列的几种情况中的若干种：

1. 理论成果不符合现实，或者无法解释实验现象，或者对实验结果的预言与实际不符。
2. 理论仅依靠于非常局限的知识来源，并辅以自己的幻想。
3. 理论与公认的科学知识无法兼容。
4. 理论无法证伪。
5. 理论无法用于预言实验结果。

诚然，并非所有反直觉的理论就一定不是科学。但科学是可证伪的，只要有人给出的理论比现有的科学更加能解释已有的实验现象并更能预言未来的实验结果，那么哪怕这个理论如何反直觉，都会被公认为新的科学。但是在日复一日、年复一年的“证明”、“推翻”的信息传播的轰炸下，人们的情绪逐渐麻木，“民科”一词在网络语境下也彻底变成了贬义词。

虽然杨淯而至少从高中开始就在不断编写各种各样的“书籍”、教程等，但遗憾的是，杨淯而在相关的文章中展现了自己在计算机基础理论方面的知识的严重缺失与自我幻想。因此著作等身的杨淯而在一定程度上可视为民科。

以下是杨淯而的“书籍”中的错误与不符合现有计算机知识体系的内容的节选（为方便表达，将《Windows 编程基础案例——用 C 语言、python、[cmd / vbs] 共同谱写 Windows 美好》[2] 简称为《基础案例》，将《Windows 编程中高级案例——用数据结构、Windows Hook、数论共同谱写 Windows 美好》[3] 简称为《中高级案例》，《指令・代码》[1] 保留原名）：

1. 《基础案例》“第零例 打印 Hello World”的“C 语言”一节中，代码注释中显示文件名为 `ConsoleApplication32.cpp`，包含的头文件为 `#include <iostream>`，代码的输出语句的行为 `std::cout << "Hello World!\n";`。
然而实际上，`.cpp` 文件扩展名意味着这是一个 C++ 文件，而 `iostream` 头文件也是 C++ 的头文件，`std::cout` 也是 C++ 的输出语句，这三个片段都指示着这是一个 C++ 的代码而非 C 语言的代码。

2. 《基础案例》除“第零例 打印 Hello World”与”第八例～二分法求 101 的八次根“与四个”项目“(不包含 C 语言代码) 之外，其他”例“中的 C 语言代码中的包含头文件的语句均为使用双引号的 `#include "stdio.h"` 格式，而非使用尖括号的 `#include <stdio.h>` 格式。

3. 《基础案例》“第一例 编程规范化”的“C 语言”一节中使用了大量的 `system` 函数。
《基础案例》开篇描述“C 语言环境：VS2019”。在对应的帮助文档中，`system` 函数的解释为：

   > 系统函数查找命令解释器，它通常是 Windows NT 操作系统中的 CMD.EXE 或 Windows 中的 COMMAND.COM。系统函数随后将自变量字符串传递到命令解释器。[4]

   也即，该 C 语言代码中大部分其实只是加了一层调用的“cmd”，严格来说并不能算真正意义上的使用 C 语言实现的内容。

4. 《基础案例》“第一例 编程规范化”的“cmd”一节中说：

   > 注意: 批处理中不区分大小写, 但使用大写运行速度较快。

   然而不管是从理论还是实际上，批处理文件中的命令的大小写对运行速度的影响是可以忽略不计的。

   - 从理论上说，由于 Windows 命令提示符确实是使用大写命令的，小写命令需要做大小写转换，但是转换的时间对于整个脚本运行的时间来说可以忽略不计。
   - 从实际上说，本文编写组成员在两台设备上的测试结果如下：

Table 1: 运行 100000 次 ECHO A 的耗时对比

$0.1\mu s$	第一次	第二次	平均	比例
小写	216959652	214635699	215797675	99.14
大写	216759705	218563371	217661538	100

$0.1\mu s$	第一次	第二次	第三次	平均	比例
小写	559739693	547107260	553989968	553612307	101.82
大写	551968566	540061682	539131738	543720662	100

可见，杨淯而说的这一句话并非正确。另外，本文编写组成员认为，在这里考虑由于命令大写或小写导致的 microbenchmark 问题，其实是一种着眼于小处优化而忽略了其他更大层次的优化的行为。

5. 《基础案例》“第三例 变量引用”的“C 语言”一节有如下片段代码：

```
#include "stdio.h"
#include "stdlib.h"
#include "math.h"//引入 math.h。
void error()//相同代码放一起，避免繁琐。
```

```
    //注意下面不要写作“0<b<=12”，否则输入大于 12 的数会出现“1<=12”从而返回 true。
    if (not (0 < b && b <= 12))
    {
        error();
        goto b;
    }
```

。然而实际上，查阅 MSVC 文档可知，not 函数需要标头 iso646.h[5]，因而因此该程序无法通过编译。

6. 《基础案例》“第七例 简易病毒”、“第十一例 十进制转换”、“第十三例～求逆元”中的“C 语言”一节均包含了 while (true)。然而实际上，true 并非 C 语言的关键字，是需要自己定义的一个值。因此该代码无法通过编译。

7. 《基础案例》“第十一例 十进制转换”中的“C 语言”一节有如下片段代码：

```
        _itoa_s(n % k, m, 10);
```

```
            _itoa_s(n % k, u, 10);
            strcat_s(u, m);
            strcpy_s(m, u);
```

，“第十五例 语言相生”的“C 语言”一节有如下片段代码：

```
17      _itoa_s(lt.tm_year + 1900, yt2, 10);
18      strcat_s(yt1, yt2);
19      strcat_s(yt1, yt3);

21      _itoa_s(lt.tm_mon, yt5, 10);
22      _itoa_s(lt.tm_mday, yt7, 10);
23      strcat_s(yt4, yt5);
24      strcat_s(yt4, yt6);
25      strcat_s(yt4, yt7);
26      strcat_s(yt4, yt8);
```

然而，这里对 _itoa_s 与 strcat_s 安全函数的使用是错误的。根据 MSDN 上的文档，_itoa_s 函数的原型 [6] 为

```
errno_t _itoa_s( int value, char * buffer, size_t size, int radix );
```

，strcat_s 函数的原型 [7] 为

```
errno_t strcat_s(
    char *strDestination,
    size_t numberOfElements,
    const char *strSource
);
```

，strcpy_s 函数的原型 [8] 为

```
errno_t strcpy_s(
    char *dest,
    rsize_t dest_size,
    const char *src
);
```

，与杨淯而的程序明显无法对应（倒数第二个参数均被忽略），这明显是错误地把一系列非安全函数的参数直接套用到安全函数上而忽略了两类函数的参数的不同的结果，反而使得安全函数不再“安全”。事实上该程序无法编译通过。

8. 《编程基础》“第十六例 获取管理员权限”的“三、cmd”一节中，杨淯而在一个描述

2.2 “第四代”反病毒引擎

2.3 多图杀猫

“多图杀猫”是在中文互联网发展早期广为流传的用语之一。二十一世纪初，上网需要使用俗称“猫”的硬件设备“调制解调器”，由于网速和硬件性能极其有限，打开含有大量图片的网页会出现计算机长时间卡顿甚至死机、网络中断等严重问题。

虽然在今天“多图杀猫”这个词已经成为了历史的一部分，即使是使用手机连接移动数据网络，其性能和速度对于上网来说也是绰绰有余，但是加快载入速度、减少加载网页（至少是在 First Contentful Paint 之前）需要载入的数据量和同时发起的请求数量仍然是前端性能优化的重要方向。

> FCP measures how long it takes the browser to render the first piece of DOM content after a user navigates to your page. Images, non-white `<canvas>` elements, and SVGs on your page are considered DOM content; anything inside an iframe isn't included. This table shows how to interpret your FCP score:
>
> | FCP time (in seconds) | Color-coding | FCP score |
> |---|---|---|
> | 0–2 | Green (fast) | 75–100 |
> | 2–4 | Orange (moderate) | 50–74 |
> | > 4 | Red (slow) | 0–49 |
>
> [9]

参考翻译：

> 首次内容渲染时间标记了渲染出首个 DOM 节点的时间。图片、非空白的 `<canvas>` 和 SVG 矢量图形也被视为 DOM 节点，但是不包括嵌入在框架（`<iframe>`）中的网页内容。下表展示了对首次内容渲染时间的评价标准：
>
> | 首次内容渲染时间（秒） | 评价 | 评分 |
> |---|---|---|
> | 0–2 | 快速 | 75–100 |
> | 2–4 | 适中 | 50–74 |
> | > 4 | 缓慢 | 0–49 |

遗憾的是，杨淯而在编写个人网站时对前端性能毫不在意，结果在高速网络早已普及的 2020 年再次闹出了“多图杀猫”的笑话。

在杨淯而个人网站的“英德”页面 [10] 中，杨淯而以图文并茂的形式抒发了对自己家乡的热爱之情。“英德”的主页上同时加载了 18 张图片，虽然一般的网站加载的图片数量可能会远远超过这个数目，但如果 18 张图片中有数张大小为 MB 级的大图，情况就变得不一样了。

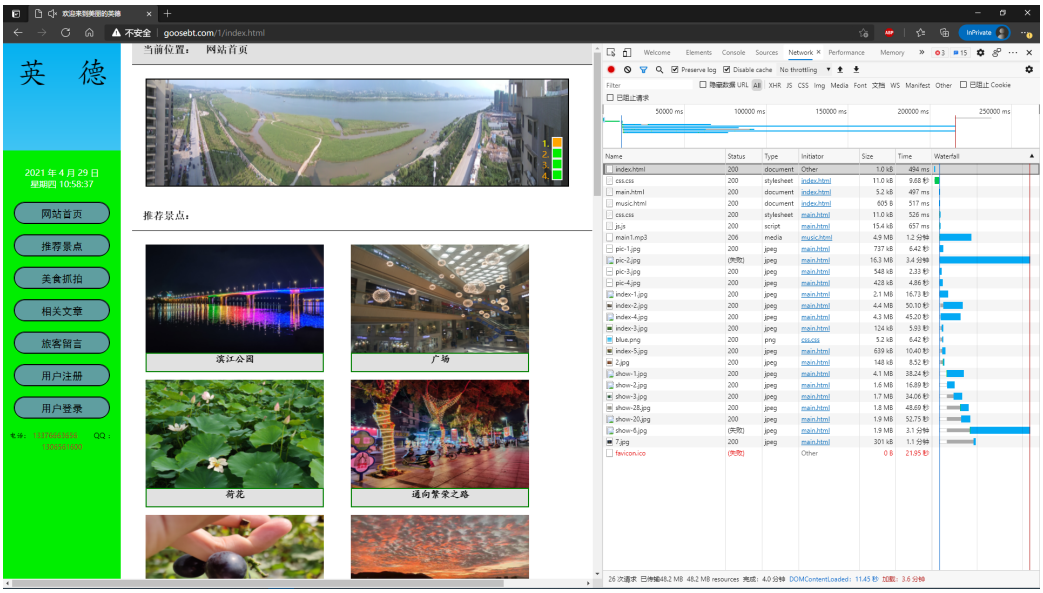

Figure 1: 使用浏览器开发者工具检查“英德”主页的网络加载情况

根据本文编写组成员在打开“英德”主页时浏览器开发者工具的记录，足足过去了 4.0 分钟浏览器才完成了网页的加载，数据量高达 48.2 MB。即使是只考虑约为 10 秒的首次内容渲染时间（时间轴的蓝线部分），按照上述的 Google 的网页性能评价工具 PageSpeed Insights 的定义和评价标准也是极其缓慢的，而对于 4.0 分钟的完成加载时间，本文编写组成员认为即使是“龟速”也不足以形容。

图片的原始高度均超过了 2000px，在网页上显示的高度却只有 200px。即使考虑到“图片资源应按照 2 倍分辨率准备以在移动端保持清晰”的规则，加载尺寸如此巨大的图片仍然是对流量和时间的极大浪费，更何况杨淯而使用的服务器带宽上限仅有 1 MB/s。最极端的是页面上轮播的全景照片之一 `img/pic-2.jpg`，尺寸为 21184x3680，大小为 18.75 MB，任何一个稍有经验的前端开发者都不会在没有特殊理由的情况下将如此大小的图片直接添加到网页上加载。

同时加载多张大图不仅挤满了带宽，甚至还严重影响了服务器的可用性。在本文编写组成员打开“英德”进行测试后，接下来超过半小时的时间杨淯而的网站都处于完全无法访问的状态，再次上演了“多图杀猫”的场景——除了这里被“杀”的是服务端而不是客户端的“猫”。

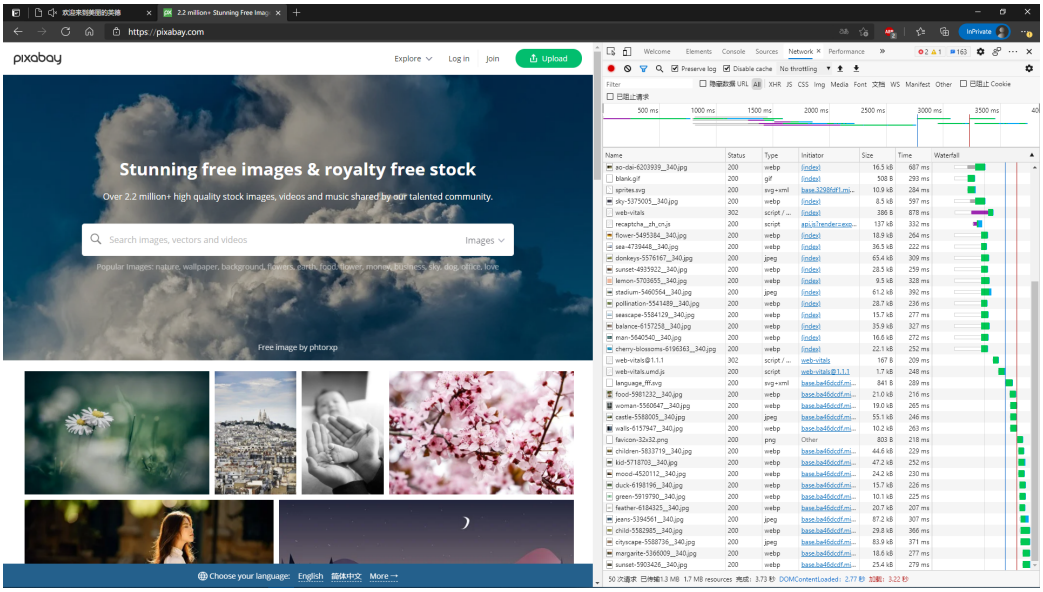

Figure 2: 使用浏览器开发者工具检查 Pixabay 主页的网络加载情况

以同样是在主页加载了大量图片的免版权图片网站 Pixabay[11] 主页作为对比，共加载图片三十多张（已完全下滑以完成所有懒加载），而总数据量仅有 1.3 MB，首次内容渲染时间不到 3 秒，完全加载用时 3.73 秒（考虑到商业网站使用的高速 CDN 是个人网站不一定具备的条件，这个指标仅供参考），其差距一目了然。

读者可能会认为杨淯而是有意而为之，就像 GitHub 上的 XP.css 项目 [12] 或清华大学开源镜像站 2020 年的愚人节玩笑“将镜像站业务转移到 Windows 95 计算机上”[13] 那样。Retrocomputing 文化（常见的中文翻译有“复古计算”、“古董电脑”等）的爱好者们会在当代收集并使用旧的电脑硬件和软件，致力于保存和传播旧时代的数字文化（参见“古董电脑 101：复古计算（Retrocomputing）简介”[14]。但据本文编写组所知，杨淯而并不是 Retrocomputing 的爱好者。

3 杨淯而的学术不端

3.1 “广撒网，多敛鱼”

2020 年 11 月，杨淯而同时携四个项目参与了信息科学技术学院/网络空间安全学院举办的“第三届吴渔夫学术科技创新竞赛”[17, 18]。这四个项目分别是 [15]*[原文如此]*：

1. 基于 resnet34 的乳腺癌病理组织切片识别系统 *（优胜奖）

2. 基于机器学习的第四代反病毒软件 *（优胜奖）

3. 基于 nginx 和 apach2 框架的服务管理系统 *（二等奖，网络投票“最具启发”奖）

4. 基于 Ngnix 和 apache2 框架的康养小镇运营平台（三等奖）

*标有 * 的项目表示杨淯而系该项目的负责人。*

单人同时参与四个项目并担任其中三个项目负责人实属罕见，不过这里暂且理解为杨淯而本人科研能力一流，效率极高。由于本文编写组对机器学习了解甚少，因此不予对“切片识别系统”进行评价。

对于“第四代反病毒软件”，杨淯而本人提供的介绍是：

> 由于传统反病毒扫描技术——特征值扫描技术已无法适用于当今病毒传播、爆发迅速以及变种多样的反病毒局势，在微点提出第三代反病毒技术——主动防御技术后，为降低误报，同时弥补对动作较小的反病毒技术缺陷，我们亟需第四代反病毒引擎来支持反病毒的进行。
>
> 项目中，C/C++ 主要负责 Hook，在 Win 32 中，我们常采用内核注入的方式进行监控，但在 Win 64 中，如果对所有进程进行注入，计算机性能势必会大打折扣，因此，我们从文件的读写上加以防护，涉及文件读、写、删除、执行时，利用文件过滤驱动，进行防御，并将文件路径交由 python，由 python 完成机器学习对反病毒的学习。[17]

本文编写组成员在 Google 上以“第三代反病毒”为关键词进行搜索，提及“第三代反病毒技术”等字样的搜索结果既有 2008 年由东方微点推出的“微点主动防御软件”的介绍 [19]，也有 2001 年由金山软件推出的“金山毒霸 2001”的介绍 [20]，因此本文编写组认为这里提及的所谓“第……代反病毒软件”仅是一个商业概念，并没有什么特别的技术含义。

在反病毒软件中利用机器学习确实是安全界研究的新方向之一，目前主流安全厂商均在旗下反病毒产品中引入了机器学习技术。这些厂商的机器学习反病毒引擎均是部署在算力强劲的云端，客户端将扫描的可疑文件发送到云端进行分析。

> 最后，在实际的生产过程中，RDM+ **通过云服务的方式对外提供服务**，不仅可以使用传统的文件哈希来抑制误报，还使用特征向量的哈希值来直接遮蔽误报，为下一次预测模型更新争取时间。[21]
>
> 到了云引擎时代，比如某个客户端发现可疑样本时，**将样本发送到云端样本分析集群里进行分析跑测**，然后将分析的结果形成特征库再下放到全网客户端，形成一个互联网病毒样本自动处理中心。[22]

而根据项目介绍，“第四代反病毒软件”是直接在客户端运行 Python 进行机器学习。众所周知，杀毒软件在扫描时过高的 CPU 占用会严重影响用户正常的使用体验，机器学习恰恰又涉及大量对计算机性能和算力有较高要求的计算任务。两者相结合，且不说反病毒效果如何（杨淯而既没有发布演示软件的下载，也没有展示使用多种病毒样本的测试结果），对计算机算力的大量消耗恐怕足以使这个反病毒软件本身成为病毒。

杨淯而后来以项目成员的身份，将项目改名为“融合主动防御与二进制图像学习的反病毒软件”继续参加学院的“2021 年吴渔夫创新创业竞赛”，在答辩环节被评委提问：

评委: 你这个深度学习是跑在云端还是在终端?
负责人: 终端。
评委: 深度学习哦？我本来都嫌杀毒软件和安全软件太多了，你还搞个深度学习上来，我的机子还跑不跑了？
负责人: 但是我们可以在终端部署软件，资源占用是比较少的。
评委: 你确定?
负责人: 不过我们后期……
评委: 深度学习，小也小不到哪里去的咧，而且你必须要二十四小时运行吧，你持续地在跑一个深度学习的东西，不要说你的 PC 跑不动了，我服务器都不一定跑得动。
负责人: 啊，这个问题我们后期会考虑一下，因为我们还没有做到这一部分。

3.2 关门大弟子

3.3 MD5 破解实用化第一人

在数学与计算机的交界处，有一类函数 H 满足这样的性质：

1. 对于一个 $y = H(x)$，x 的长度是任意的。
2. 对于一个 $y = H(x)$，y 的长度是固定的。
3. 对于一个 $y = H(x)$，已知 x 求出 y 的过程对于计算机来说是容易的。
4. 对于一个 $y = H(x)$，已知 y 求出 x 是计算上不可行的。
5. 对于一个 x_1，求出一个 x_2 以满足 $H(x_1) = H(x_2)$ 是计算上不可行的。
6. 任意找到一对满足 $H(x_1) = H(x_2)$ 的 x_1 和 x_2 是计算上不可行的。
7. 对于不同的 x，满足 $y = H(x)$ 的 y 看起来是随机的。

这一类函数被成为“杂凑函数”，也经常以其英文 Hash 音译为“哈希函数”。它的性质可以用于简单地辨别文件是否相同。

4 杨淯而在日常生活中的破坏行为

4.1 “云”音绕梁，三日不绝

4.2 时间管理大师

4.3 凡尔赛人

5 参考资料

[1] 杨淯而. 指令・代码 [M]. 广东: 杨淯而, 2019

[2] 杨淯而.Windows 编程基础案例——用 C 语言、python、[cmd / vbs] 共同谱写 Windows 美好 [M]. 广东: 杨淯而, 2020

[3] 杨淯而.Windows 编程中高级案例——用数据结构、Windows Hook、数论共同谱写 Windows 美好 [M]. 广东: 杨淯而, 2020

[4] Microsoft.system Function[EB/OL].Microsoft, 2022(2022-05-18)[2022-07-07].https://docs.microsoft.com/en-us/cpp/c-language/system-function?view=msvc-160.

[5] Microsoft.not[EB/OL].Microsoft, 2022(2022-05-02)[2022-07-10].https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/not?view=msvc-160

[6] Microsoft._itoa_s, _itow_s functions[EB/OL].Microsoft, 2022(2022-05-02)[2022-07-10].https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/itoa-s-itow-s?view=msvc-160.

[7] Microsoft.strcat_s, wcscat_s, _mbscat_s, _mbscat_s_l[EB/OL].Microsoft, 2021(2021-04-08)[2022-07-10].https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/strcat-s-wcscat-s-mbscat-s?view=msvc-160

[8] Microsoft.strcpy_s, wcscpy_s, _mbscpy_s, _mbscpy_s_l functions[EB/OL].Microsoft, 2022(2022-05-28)[2022-07-10].https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/strcpy-s-wcscpy-s-mbscpy-s?view=msvc-160

[9] Google.First Contentful Paint[EB/OL].Google,2019(2019-10-10)[2021-04-29].https://web.dev/first-contentful-paint/.

[10] 杨淯而. 欢迎来到美丽的英德 [EB/OL]. 杨淯而,2020(2021-02-10[2021-04-29].http://goosebt.com/1/index.html.

[11] Pixabay.Pixabay - 2.2 million+ Stunning Free Images to Use Anywhere[EB/OL].Pixabay,2010(2021-04-29)[2021-04-29].https://pixabay.com.

[12] botoxparty.botoxparty/XP.css: A CSS framework for building faithful recreations of operating system GUIs.[EB/OL].GitHub,2020(2020-12-21)[2021-04-29].https://github.com/botoxparty/XP.css.

[13] 清华大学 TUNA 协会. 清华大学开源软件镜像站 | Tsinghua Open Source Mirror[EB/OL]. 清华大学,2020(2020-04-01)[2021-04-29].https://web.archive.org/web/20200401001002/https://mirrors.tuna.tsinghua.edu.cn/.

[14] 蓬岸. 古董电脑 101：复古计算（Retrocomputing）简介 [EB/OL]. 知 乎,2018(2018-02-12)[2021-04-29].https://zhuanlan.zhihu.com/p/41742029.

[15] 学院团委. 牢记嘱托·榜样 | 2020 年第三届吴渔夫学术科技创新竞赛获奖名单公布 [EB/OL]. 微信: 暨大信息学生学习与发展,2020(2020-11-17)[2021-04-28].https://mp.weixin.qq.com/s/MsMwQaMBQlNkRpvYksSFSQ.

[16] 学院团委. 牢记嘱托 | 学无止境、勇攀高峰——记第三届吴渔夫学术科技创新竞赛圆满成功! [EB/OL]. 微信: 暨大信息学生学习与发展,2020(2020-11-15)[2021-04-28].https://mp.weixin.qq.com/s/pBSB_PlNCtRqmzIr6CyeGQ.

[17] 学院团委. 预告 | 吴渔夫学术科技创新竞赛项目展示上篇 [EB/OL]. 微信: 暨大信息学生学习与发展,2020(2020-11-10)[2021-04-28].https://mp.weixin.qq.com/s/7JfWZnPBKE9-RhOsYUvwew.

[18] 学院团委. 预告 | 吴渔夫学术科技创新竞赛项目展示下篇 [EB/OL]. 微信: 暨大信息学生学习与发展,2020(2020-11-10)[2021-04-28].https://mp.weixin.qq.com/s/7pa4TxSiLeOEk2dqvvTCDQ.

[19] 东方微点信息技术有限责任公司. 第三代反病毒产品－微点主动防御软件概述 [EB/OL]. 微点,2008(2018-09-20)[2021-04-28].http://shop.micropoint.com.cn/product/index.htm.

[20] 新浪科技. 第三代杀毒软件金山毒霸 2001 完全写真 (附图)[EB/OL]. 新 浪,2001(2001-08-21)[2021-04-28].https://tech.sina.com.cn/s/n/2001-08-21/81447.shtml.

[21] 瑞星.2017 国际反病毒大会 瑞星已在反病毒领域全面使用人工智能技术 [EB/OL]. 瑞星,2017(2017-11-09)[2021-04-28].https://it.rising.com.cn/dongtai/19177.html.

[22] 朱立娜. 机器学习和未知样本检测之云中的反病毒引擎 [EB/OL].IT168,2017(2017-10-21)[2021-04-28].http://cloud.it168.com/a2017/1021/3175/000003175452.shtml.