2023 年第五届

全球校园人工智能算法精英挑战赛

技 术 报 告

赛题二：电力大模型问答挑战赛

参赛队名：GPT

二〇二三年十二月

目录

一、 赛题背景说明 3

1.1 赛题背景 3
1.2 赛题说明 3

二、 赛题实验结果 3

三、 主要技术路线及实现方式 4

3.1 数据采集与处理 4
3.1.1 数据采集 4
3.1.2 数据处理 5
3.1.3 数据说明 5
3.2 技术路线 5
3.2.1 知识库检索增强生成技术 5
3.2.2 GLM2-6B 大模型微调 6
3.2.3 PROMPT 与问答策略 9

四、 遇到的问题及解决方法 10

4.1 遇到的问题 10
4.2 解决方法 10

五、 主要创新点 11

六、 复现过程 11

6.1 实验环境 11
6.2 运行方法 11

七、 参考文献 13

八、 附录 13

一、赛题背景说明

1.1 赛题背景

当前大语言模型在众多任务上都展现出强大的问答能力，但在知识密集型领域往往表现欠佳，如何提升大模型在知识密集领域的能力是一个有趣的问题。电力是一个典型的知识密集型领域，同时涉及电子、物理、化学、数学等学科知识，是一个良好的衡量大模型知识密集型能力的场景，一个优秀的电力问答大模型需要同时掌握多个学科背景知识，并具备在此基础上进行归纳和推理的能力。

1.2 赛题说明

本次比赛要求参赛选手以 ChatGLM2-6B 模型为核心制作一个问答系统，回答电力领域相关的问题。题目类型包含单选题、多选题和自由问答。本次比赛所提供数据均为测试集，用于测试大模型的能力。比赛不允许使用其他的大语言模型，但可以使用其他大语言模型生成的数据，也可以使用其他公开访问的外部数据来微调模型。选手需要严格使用大模型进行作答并提交答案，严禁人工作答，违者取消参赛资格。

二、赛题实验结果

利用官方提测，模型对于 B 榜数据最终评分 48.131。

初赛B榜 初赛A榜

排名	队伍编号	队伍名称	学校名称	所在省份	评分	最优成绩提交时间
1	bdc2340253	圣主堆	同济大学	上海市	67.680	2023-12-02 11:03
2	bdc2340267	啥也不会	华中科技大学	湖北省	56.100	2023-12-02 15:55
3	bdc2340251	SCI今天吃什么	东华大学	上海市	55.615	2023-12-02 11:06
4	bdc2340258	wuwuwubudale	中国石油大学（...	北京市	53.469	2023-12-02 13:42
5	bdc2340265	GPT	华中科技大学	湖北省	48.131	2023-12-02 19:04
6	bdc2340257	吾六武	扬州大学	江苏省	47.889	2023-12-02 18:06
7	bdc2340336	积极向上队	燕山大学	河北省	47.258	2023-12-02 23:52
8	bdc2340294	人工智能精英队	重庆科技学院	重庆市	46.248	2023-12-02 23:47
9	bdc2340361	TJTeam	同济大学	上海市	45.016	2023-12-02 13:08
10	bdc2340296	ELAN	中国科学技术大学	安徽省	42.770	2023-12-02 12:40

图 1　B 榜最终结果排名

三、 主要技术路线及实现方式

3.1 数据采集与处理

3.1.1 数据采集

训练数据集主要包含四个内容：

- 学科教材。学科教材中的文件来自官方提供。
- 国标文件。受注册电气工程师执业资格专业考试考纲中的参考国标文件启发，我们收集了所有的这些文件，并转换为可提取文字的 PDF 格式。
- 说明手册。为了提高知识库的容量，我们对“考试题库”中的难题的出处、参考文件进行检索收集，并转换为可提取文字的 PDF 格式。
- 考试题库。为了提高 chatglm2-6b 的推理能力，我们在题库网站上，对注册电气工程师执业资格专业考试题目进行爬取，并处理为{"question":"...","answer":"..."}的格式，收录在 two.json 中。

训练数据集的文件构成具体如下：

```
|── 训练数据集
|    ├── 学科教材
|        ├── 《电工学》（第六版）上册+秦曾煌主编.md
|        ├── 《电工学》（第六版）下册+秦曾煌主编.md
|        ├── ...
|        └── 注册电器工程师专业知识 1-513 复习教程.md
|    ├── 国标文件
|        ├── DLT620.pdf
|        ├── DLT621.pdf
|        ├── ...
|        └── JGJ242.pdf
|    ├── 说明手册
|        ├── 非直接接地系统单相接地故障选线方法综述.pdf
|        ├── 高层民用建筑消防安全管理规定.pdf
|        ├── 工业与民用供配电设计手册.pdf
|        └── 供电服务标准.pdf
|    ├── 考试题库
|        └── two.json
```

图 2 训练数据集文件

3.1.2 数据处理

在得到“训练数据集”的基础上，我们对“学科教材”和“考试题库”进行了一步的处理，具体如下：

- 对学科教材，为了方便 ChatGLM2-6B 的微调，我们使用 GPT-3.5，令其对学科教材的内容进行整合，整合为{"question":"...","answer":"..."}的格式。具体地，我们使用了如下的 prompt

  ```
  prompt = '''请将上列信息转换成填空问答题的形式，包含全部信息，填空的位置用中文括号（ ），问答题格式为"问题：{问题内容}\n 答案：{答案内容}\n\n"，同时问题和答案后面绝对不能添加数字表示顺序,问题内容和答案内容中间绝对不能换行'''
  ```

得到的文件整合后为 three.json。

对考试题库，首先进行了数据清洗，对爬取的格式不正确的数据进行删除、更改。

3.1.3 数据说明

在最终的实现中，对于上述的数据，在“知识库检索增强生成技术”中，我们主要使用了国标文件、说明手册、学科教材、two.json 作为知识库来进行检索增强生成；在“微调”中，我们主要使用了 two.json 来提高大模型的推理能力。

3.2 技术路线

3.2.1 知识库检索增强生成技术

为了给大模型提供与题目相关的有用的知识，我们使用了检索增强生成(RAG)技术。RAG 将输入问题与知识库中的数据进行匹配，通过计算相似度获取最相关的知识片段，然后，用提取到的知识片段来增强大模型对问题的回答生成过程。

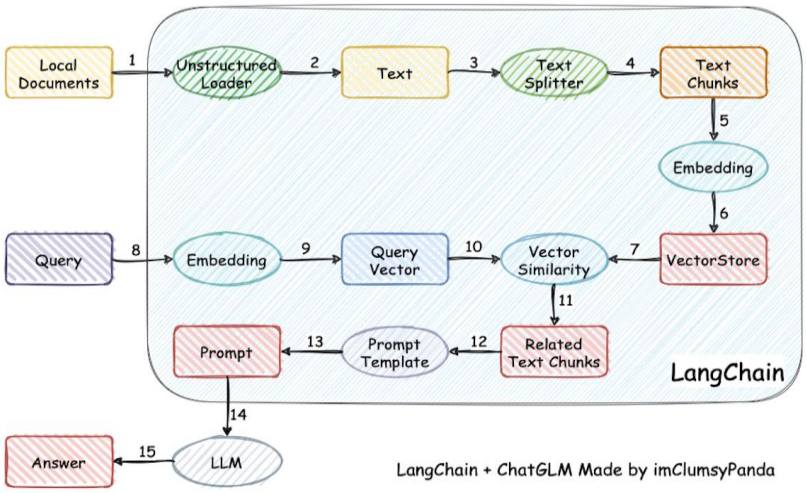

图 3 RAG 技术原理图

具体地，我们使用了 Facebook AI Similarity Search 中的 indexFlatL2 来衡量相似度，降低空间占用加快检索速度；使用国标文件、说明手册、学科教材、two.json 作为知识库，每次训练使用 RAG 技术获取最相关的知识片段，然后形成 prompt，增强大模型对问题的回答生成过程。

3.2.2 GLM2-6B 大模型微调

为了使大模型适应特定的任务或领域（本比赛中指电力领域），我们使用微调的方法，即通过在原始模型的基础上，使用新的、相对较小的数据集进行额外的训练，调整 GLM2-6B 的参数以适应比赛任务，从而提高性能。

对于 GLM2-6B 的微调策略，我们通过控制变量法实验对比了 Freeze、P-Tuning、P-Tuning-V2 以及 Lora 四种微调方法，最终得出 P-Tuning-V2 在该微调中表现最优的结论。微调数据主要来源于教材和题库（实验结果中对题库的微调效果更优），其数据 json 格式示例如下：

```
"0": {
        "question": "电工学是什么？",
        "answer": "电工学是研究电工技术和电子技术的理论和应用的技术基础课程。
"
},
```

初赛数据集A成绩

序号	评分	提交时间	提交历史
1	34.863	2023-11-29 22:54	下载
2	35.924	2023-11-28 10:25	下载
3	33.807	2023-11-28 10:14	下载
4	36.619	2023-11-26 21:36	下载
5	38.080	2023-11-23 10:22	下载
6	{"success":false,"team_id":null,"score":0,"Error_message":"utf-8"}	2023-11-23 10:15	下载
7	{"success":false,"team_id":null,"score":0,"Error_message":"上交文件含答案个数为328，而应回答320个题目，请检查你上交的答案。"}	2023-11-23 09:43	下载
8	37.680	2023-11-22 22:04	下载
9	37.680	2023-11-22 22:04	下载
10	37.680	2023-11-22 10:55	下载
11	17.427	2023-11-21 09:14	下载
12	16.427	2023-11-21 09:13	下载
13	25.313	2023-11-21 09:13	下载
14	23.990	2023-11-20 22:44	下载
15	23.990	2023-11-20 22:44	下载
16	23.990	2023-11-20 22:42	下载
17	23.262	2023-11-20 22:42	下载
18	23.789	2023-11-20 22:40	下载
19	23.989	2023-11-20 22:40	下载
20	21.972	2023-11-19 23:05	下载
21	21.920	2023-11-19 23:03	下载
22	22.320	2023-11-19 23:02	下载
23	24.094	2023-11-18 22:19	下载
24	24.094	2023-11-18 22:18	下载
25	24.340	2023-11-17 19:42	下载

图 4 初赛 A 榜提交记录图

1. Freeze

Freeze 方法，即参数冻结，对原始模型部分参数进行冻结操作，仅训练部分参数，以达到在单卡或多卡，不进行 TP 或 PP 操作就可以对大模型进行训练。在 A 榜提交记录中，序号 13 次提交采用该方案，评分 23.990。

2. P-Tuning、P-Tuing-V2

P-Tuning 是一种针对于大模型的 soft-prompt 方法，通过在训练过程中引入一些额外的损失来优化模型的表现。事实上，P-Tuning 仅对大模型的 Embedding 加入新的参数。

P-Tuning-V2 同样采用前缀微调的做法，但移除了重参数化的编码器并且针对不同任务采用不同的提示长度。不同于 P-Tuning，P-Tuning-V2 对大模型的 Embedding 和每一层前都

加上了新的参数。实验中 P-Tuning-V2 的设置如下。

```
pre_seq_len=128        #输入长度
learning_rate=2e-2     #学习率
quantization_bit=4     #4 位量化
per_device_train_batch_size=16      #训练批次大小
gradient_accumulation_steps=1       #一次参数更新前梯度累积的步骤数
```

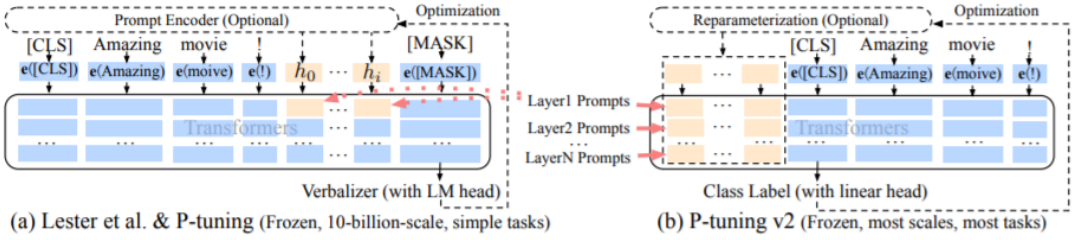

图 5　PT 微调原理示意图

在 A 榜提交记录中，序号 11 次提交采用原始 PT 方案，评分 17.427。序号 5、9、10 次提交采用 P-Tuning-V2，最高评分 38.080。

3. Lora

Lora 方法通过在模型的权重和激活之间引入一个低精度的线性层，来减少模型的计算复杂性和内存占用。即对指定参数（权重矩阵）并行增加额外的低秩矩阵，并仅训练额外增加的并行低秩矩阵的参数。当“秩值”远小于原始参数维度时，新增的低秩矩阵参数量也很小。

在 A 榜提交记录中，序号 1、2、3 次提交采用 Lora 方案，评分在 34 左右。

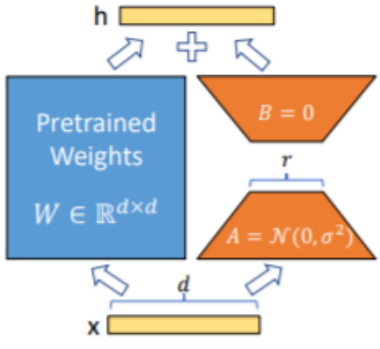

图 6　Lora 微调原理示意图

4. 微调分析

表 1 微调评分记录表

微调方案	控制变量下的提测评分
Freeze	23.990
P-Tuning	17.427
P-Tuning-V2	**38.080**
Lora	34.863

在本次比赛实验中，我们发现 P-Tuning-V2 策略的效果最好，这可能是因为 P-Tuning-V2 策略对大模型的 Embedding 和每一层前都加上了新的参数，引入了更复杂的损失函数，使得模型能够更好地学习数据的分布。此外，P-Tuning-V2 能针对不同任务采用不同的提示长度，又因为提示长度在提示优化方法的超参数搜索中起着核心作用，所以使得 P-Tuning-V2 有着更好的优化性能。

3.2.3 Prompt 与问答策略

对于问答任务来说，良好的 Prompt 和合理的问答策略也是能够很好解决问题的关键，为了获得良好的回答效果，我们在 Prompt 和问答策略中采用了以下的方法：

1. 将多分类问题转化为二分类问题

   我们首先对单选题测试了使用问答模型进行四分类的任务，标准格式为“请问{问题}，下列选项中哪一个填到括号中正确\n{A 选项}{B 选项}{C 选项}{D 选项}”，经过测试和验证，我们发现选项的顺序对于题目回答的正确性有很大的影响，会产生误判，因此我们最终采用二分类的形式，通过将选项内容填写到括号中，对问题的每个选项询问是否正确来进行判断。

2. 通过角色扮演形式进一步调出模型内的知识

   我们在实践过程中发现，大模型中原本就有很多电力相关的基础知识和常识，但是由于模型的参数量过少，又要保持模型的泛化性，ChatGLM2-6B 模型的训练采用了很多的知识压缩，参数压缩等方法，导致很难调出某一领域的相关知识。我们查阅论文发现可以通过让大模型进行角色扮演的形式，激活相关专业领域的知识，因此我们首先在 Prompt 中赋予模型一个身份“现在你是一个电力相关领域的专家”，最终实现了更好的效果。

3. Prompt 调整与实现

我们通过实际实验发现，ChatGLM2-6B 模型对于 Prompt 很敏感，仅仅改变一些看似无关紧要的文字，也可能对模型的效果产生巨大的影响，因此我们对于 Prompt 进行了精心的设计和反复的调整，通过对于单个语句的多种表达进行测试，最终选择效果最好的模型。

4. 问答策略 1

ChatGLM2-6B 模型微调需要质量较高的数据集，在本次任务中我们的微调都是基于单论对话进行微调，不可避免会让大模型丧失一定的上下文理解能力，因此我们的策略是将问答问题询问微调后的 ChatGLM 模型，进一步将此模型的回答和 RAG 检索得到的结果作为一个原模型的输入，进一步得到结果，这样同时利用了微调模型的微调知识和知识库的相关知识，也充分发挥了模型上下文的推理能力，保证最终良好的效果。

5. 问答策略 2

由于 ChatGLM2-6B 模型本身参数量的限制，很容易出现幻读，复读，错读等现象，因此对于不同种类的题目我们采用了不同的策略。对于单选题采用问答三次，最终选择出现次数最多的选项作为最终的结果；对于多选题采用问答两次取交集的方法作为最终的结果；对于问答题采用问答三次，将三次的结果拼接到一起作为最终的结果。

四、遇到的问题及解决方法

4.1 遇到的问题

- 问题 1：数据传输问题。用于微调以及知识库的数据较大，上传到运行主机需要耗费大量时间。
- 问题 2：程序所需环境冲突问题。在模型训练主机中，由于 cuda 版本等原因导致的各依赖包所需版本冲突问题。
- 问题 3：微调选择以及参数调试训练方向探索

4.2 解决方法

- 针对问题 1：在知识库构建中，我们不在每次运行的时候将所有文本放入内存中，而是通过向量库 faiss 构建知识库；在微调中，我们通过增大训练轮次，实现模型推理能力和知识融合的 tradeoff。

- 针对问题 2：查询官方文档，手动重设 requirement.txt 版本要求兼顾程序所需依赖包版本。对于 cuda 版本较低导致的 torch 版本限制问题，通过手动下载软件包安装的方式解决。
- 针对问题 3：调研不同的微调特性，选取多种方案（如 Freeze、PT、Lora 等）实验探索，对于不同参数控制变量调试更优的模型。

五、主要创新点

- 创新点 1：任务重点推断。通过控制变量的实验，比较使用教程进行微调和使用题库进行微调二者的优劣（实验结果题库微调更优），从而得知任务对选项推理能力的关注。
- 创新点 2：利用大模型优化大模型。项目实现中对教材、题库的知识解析借助了包括但不限于 ChatGPT 在内的大模型，顺应时代潮流。
- 创新点 3：精心设计的 Prompt 和谨慎的问答策略，帮助模型更好地理解任务，分析任务，回答问题。

六、复现过程

6.1 实验环境

- 系统环境：Ubuntu 20.04
- Python 环境：3.9
- Anaconda
- 显卡：8 卡 NVIDIA TESLA V100 （单卡显存 32G）
- CUDA 版本：11.6
- 安装包：请参见 chatchat_use 文件夹下的 requirements.txt

6.2 运行方法

1. 创建环境

   首先进入 chatchat_use 目录下，使用 conda 创建 llm 环境（最好不要在主环境安装，避免包冲突），同时进入环境。

```
conda create -n llm python=3.9
conda activate llm
```

2. 安装相关的依赖库（请使用清华源）

```
pip install -r requirements.txt
```

特殊说明：我们使用的 CUDA 版本为 11.6，支持的最高 torch 版本为 torch1.13.1，如果你的 CUDA 版本更低，请自行选择 torch 版本（但是不确保可以运行），如果运行上面的额依赖安装命令有问题，请先运行下面的命令行，然后删除 requirements.txt 中的前三行，再次运行上面的 pip 指令。

```
pip install torch==1.13.1+cu116 torchvision==0.14.1+cu116 torchaudio==0.13.1 --extra-index-url https://download.pytorch.org/whl/cu116
```

最终安装包列表请见附录，如有不同请自行更改。

3. 运行问答程序

程序的核心代码是 answer_json.py，读取当前路径下的 question.json 文件并回答问题，在 chatuse.py 中可以根据实际服务器环境选择显卡数量，原模型参数路径为 chatchat_use/chatglm2，微调后的模型参数位置为 chatchat_use/chatglm2/czx/ChatGLM2-6B/ptuning/output/final-answer/checkpoint-10000，知识库文件路径为 chatchat_use/vector_store。具体运行代码如下。

```
python answer_json.py
```

4. 运行结果说明

程序运行所需要的时间较长，大概需要 12-16h，所以需要耐心等待，或者自行开启多线程运行。在问答题的答案中，因为 chatglm 自身参数量过小，可能发生自陷，有可能出现答案过长（重复）的现象，如果 VScode 不支持过长文本解析，请自行删除多次重复的部分，不会影响最终的分数计算。

5. 如果在上述步骤中出现任何问题，请联系团队成员解决。

七、参考文献

[1]Zeng, A., et al. "Glm-130b: An open bilingual pre-trained model," in arXiv preprint arXiv:2210.02414, 2022.

[2]Du, Z., et al, "GLM: General Language Model Pretraining with Autoregressive Blank Infilling," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 320–335.

[3]Liu, X., et al. "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," in arXiv preprint arXiv:2110.07602, 2021.

[4]https://github.com/liucongg/ChatGLM-Finetuning

八、附录

Package	Version
accelerate	0.23.0
aiofiles	23.2.1
aiohttp	3.8.6
aiosignal	1.3.1
altair	5.1.2
annotated-types	0.6.0
antlr4-python3-runtime	4.9.3
anyio	3.7.1

appdirs	1.4.4
async-timeout	4.0.3
attrs	23.1.0
backoff	2.2.1
backports.zoneinfo	0.2.1
beautifulsoup4	4.12.2
bitsandbytes	0.41.0
blinker	1.6.3
cachetools	5.3.1
certifi	2023.7.22
cffi	1.16.0
chardet	5.2.0
charset-normalizer	3.3.0
click	8.1.7
coloredlogs	15.0.1
contourpy	1.1.1
cpm-kernels	1.0.11
cryptography	41.0.5
cycler	0.12.1
dataclasses-json	0.5.14
datasets	2.14.6
deepspeed	0.12.2
dill	0.3.7
docker-pycreds	0.4.0
effdet	0.4.1
emoji	2.8.0
et-xmlfile	1.1.0
exceptiongroup	1.1.3
faiss-cpu	1.7.4
fastapi	0.103.2

ffmpy	0.3.1
filelock	3.12.4
filetype	1.2.0
flatbuffers	23.3.3
fonttools	4.43.1
frozenlist	1.4.0
fsspec	2023.9.2
gitdb	4.0.10
GitPython	3.1.37
gradio	3.48.0
gradio_client	0.6.1
greenlet	3.0.1
h11	0.14.0
hjson	3.1.0
httpcore	0.18.0
httpx	0.25.0
huggingface-hub	0.18.0
humanfriendly	10.0
icetk	0.0.7
idna	3.4
importlib-metadata	6.8.0
importlib-resources	6.1.0
iopath	0.1.10
jieba	0.42.1
Jinja2	3.1.2
joblib	1.3.2
jsonschema	4.19.1
jsonschema-specifications	2023.7.1
kiwisolver	1.4.5
langchain	0.0.146

langdetect	1.0.9
latex2mathml	3.76.0
layoutparser	0.3.4
lxml	4.9.3
Markdown	3.5
markdown-it-py	3.0.0
MarkupSafe	2.1.3
marshmallow	3.20.1
matplotlib	3.7.3
mdtex2html	1.2.0
mdurl	0.1.2
mpmath	1.3.0
msg-parser	1.2.0
multidict	6.0.4
multiprocess	0.70.15
mypy-extensions	1.0.0
networkx	3.1
ninja	1.11.1.1
nltk	3.8.1
numexpr	2.8.6
numpy	1.24.4
nvidia-cublas-cu12	12.1.3.1
nvidia-cuda-cupti-cu12	12.1.105
nvidia-cuda-nvrtc-cu12	12.1.105
nvidia-cuda-runtime-cu12	12.1.105
nvidia-cudnn-cu12	8.9.2.26
nvidia-cufft-cu12	11.0.2.54
nvidia-curand-cu12	10.3.2.106
nvidia-cusolver-cu12	11.4.5.107
nvidia-cusparse-cu12	12.1.0.106

nvidia-nccl-cu12	2.18.1
nvidia-nvjitlink-cu12	12.2.140
nvidia-nvtx-cu12	12.1.105
olefile	0.46
omegaconf	2.3.0
onnx	1.12.0
onnxruntime	1.14.1
openai	0.28.1
openapi-schema-pydantic	1.2.4
opencv-python	4.8.1.78
openpyxl	3.1.2
orjson	3.9.9
packaging	23.2
pandas	2.0.3
pdf2image	1.16.3
pdfminer.six	20221105
pdfplumber	0.10.3
peft	0.6.1
Pillow	10.1.0
pip	23.2.1
pkgutil_resolve_name	1.3.10
portalocker	2.8.2
protobuf	3.19.0
psutil	5.9.6
py-cpuinfo	9.0.0
pyarrow	13.0.0
pycocotools	2.0.7
pycparser	2.21
pydantic	1.10.13
pydantic_core	2.10.1

pydeck	0.8.1b0
pydub	0.25.1
Pygments	2.16.1
pynvml	11.5.0
pypandoc	1.12
pyparsing	3.1.1
pypdfium2	4.23.1
pytesseract	0.3.10
python-dateutil	2.8.2
python-docx	1.1.0
python-iso639	2023.6.15
python-magic	0.4.27
python-multipart	0.0.6
python-pptx	0.6.21
pytz	2023.3.post1
PyYAML	6.0.1
rapidfuzz	3.5.2
referencing	0.30.2
regex	2023.10.3
requests	2.31.0
rich	13.6.0
rouge-chinese	1.0.3
rpds-py	0.10.6
safetensors	0.4.0
scikit-learn	1.3.2
scipy	1.10.1
semantic-version	2.10.0
sentence-transformers	2.2.2
sentencepiece	0.1.99
sentry-sdk	1.34.0

setproctitle	1.3.3
setuptools	68.0.0
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
some-package	0.1
soupsieve	2.5
SQLAlchemy	1.4.50
sse-starlette	1.6.5
starlette	0.27.0
streamlit	1.27.2
sympy	1.11.1
tabulate	0.9.0
tenacity	8.2.3
threadpoolctl	3.2.0
timm	0.9.10
tokenizers	0.13.3
toml	0.10.2
toolz	0.12.0
torch	1.13.1+cu116
torchaudio	0.13.1+cu116
torchvision	0.14.1+cu116
tornado	6.3.3
tqdm	4.66.1
transformers	4.30.2
triton	2.1.0
typing_extensions	4.8.0
typing-inspect	0.9.0
tzdata	2023.3
tzlocal	5.1

unstructured	0.10.28
unstructured-inference	0.7.10
unstructured.pytesseract	0.3.12
urllib3	2.0.6
uvicorn	0.23.2
validators	0.22.0
wandb	0.16.0
watchdog	3.0.0
websockets	11.0.3
wheel	0.41.2
xlrd	2.0.1
XlsxWriter	3.1.9
xxhash	3.4.1
yarl	1.9.2
zipp	3.17.0