

天津大学



知识图谱构建及可视化功能使用指南

编者：3023244322 蒋茜，3023244328 马佳一

资料提供：3023244327 邵玺冉，3023244338 张婉毓

（姓名不分先后，按首字母排序）

2025 年 6 月 30 日

目录

知识图谱构建及可视化功能使用指南	1
第一章 数据的采集	3
第二章 设计知识图谱的 schema	4
第三章 抽取实体和关系	6
第四章 数据处理及完善	12
第五章 构建知识图谱三元组关系	14
第六章 neo4j 数据导入及可视化	18
第七章 小结	19

第一章 数据的采集

1.1 数据采集

本次构建知识图谱的数据来源于三方面。

首先是课程要求的《人工智能知识体系》电子书中的两小节文本内容，本小组分配到的章节为 2.1 基本概念与主要术语，2.2 绪论。其次为在知网上发表的引用量较高的相关论文，如《人工智能的适应性表征认知理论_魏屹东》等。第三为相关学科书籍的节选篇章，如《06-脑科学导论（第 9 章、第 11 章节选）》等。

全部采集数据整理成文档 20 篇，共计 30 万字，具体文档列举如下：

文件资源 > ReferenceDocuments

排序查看

名称	修改日期	类型	大小
0-必选文件.docx	2025/6/30 0:21	DOCX 文档	14 KB
01-面向人工智能的语言认知：话语基础...	2025/6/26 21:22	DOCX 文档	25 KB
02-面向认知计算的智能自适应学习系统...	2025/6/26 21:48	DOCX 文档	20 KB
03-人工认知的语境建构与适应性表征解...	2025/6/26 22:08	DOCX 文档	251 KB
04-科学计量视角下认知科学发展研究.do...	2025/6/27 22:14	DOCX 文档	1,047 KB
05-论人文学科在认知科学中的作用——...	2025/6/27 22:14	DOCX 文档	1,536 KB
06-脑科学导论（第9章、第11章节选）	2025/6/27 22:14	DOCX 文档	20 KB
07-认知科学的三大基石_陈霖-1.docx	2025/6/27 22:13	DOCX 文档	38 KB
08-认知科学的兴起及其发展路径.docx	2025/6/27 22:14	DOCX 文档	507 KB
09-认知神经科学视角下的数字阅读认知...	2025/6/27 22:15	DOCX 文档	148 KB
论意识的认知神经科学研究及哲学思考.pdf	2025/6/28 22:52	PDF 文件	316 KB
人工智能的适应性表征认知理论_魏屹东....	2025/6/28 22:21	PDF 文件	1,473 KB
认知工作分析在智能交互设计教学中的应...	2025/6/28 22:33	PDF 文件	3,887 KB
认知科学的现状与发展趋势.pdf	2025/6/29 21:30	PDF 文件	255 KB
认知科学和计算机隐喻等问题_李康-1.pdf	2025/6/28 22:30	PDF 文件	138 KB
认知神经科学、人工智能与创造力.pdf	2025/6/28 22:50	PDF 文件	1,182 KB
认知神经科学研究及其哲学意义.pdf	2025/6/28 22:54	PDF 文件	289 KB
认知语言学与两代认知科学.pdf	2025/6/28 22:58	PDF 文件	430 KB
适应性表征_架构自然认知与人工认知的...	2025/6/28 22:18	PDF 文件	344 KB
知识可视化的理论与实践探索——基于...	2025/6/28 22:23	PDF 文件	897 KB

第二章 设计知识图谱的 schema

2.1 节点类型

通过将文档初步交给 deepseek 设计 schema，解析出来了主要的节点类型，并结合文章进行了一定的解释，便于之后的实体和关系抽取。共计 35 种节点类型，具体表格如下：

序号	本体类型	本体	解释
1	基础认知本体	认知	该实体类型描述的是人类获取、处理、存储和使用信息的心理过程。包括感知、记忆、思维、语言等高级心理功能。
2	基础认知本体	智能	该实体类型描述的是个体适应环境、学习知识和解决问题的能力系统。包括流体智力、晶体智力、情绪智力等多
3	基础认知本体	知识	该实体类型描述的是通过经验或教育获得的事实、信息、技能和理解的总和。包括陈述性知识和程序性知识。例
4	基础认知本体	记忆	该实体类型描述的是信息编码、存储和提取的认知过程。包括感觉记忆、工作记忆和长时记忆系统。例如：情景
5	基础认知本体	学习	该实体类型描述的是通过经验引起行为或认知结构持久改变的过程。包括联想学习、观察学习、内隐学习等形式
6	基础认知本体	语言	该实体类型描述的是人类用于交流的符号系统，包括语音、语法和语义三个层次。例如：汉语、英语、手语等自
7	基础认知本体	意识	该实体类型描述的是个体对自身存在、思想和环境的觉知状态。包括现象意识、存取意识和自我意识等维度。例
8	基础认知本体	注意	该实体类型描述的是认知资源的选择性分配过程。包括选择性注意、分配性注意和持续性注意。例如：鸡尾酒会
9	基础认知本体	感知	该实体类型描述的是感觉信息的组织和解释过程。包括视觉、听觉、触觉等多种模态。例如：格式塔知觉组织原
10	基础认知本体	思维	该实体类型描述的是心理表征的操作过程。包括概念形成、问题解决、决策制定等高级认知功能。例如：归纳推
11	认知科学分支	认知心理学	该实体类型描述的是研究人类心智活动的心理学分支学科。采用信息加工观点解释认知过程。例如：记忆的加工
12	认知科学分支	认知神经科学	该实体类型描述的是研究认知过程的神经机制的交叉学科。结合心理学和神经科学方法。例如：工作记忆的神经
13	认知科学分支	认知语言学	该实体类型描述的是研究语言与认知关系的语言学分支。强调语言使用中的概念化过程。例如：概念隐喻理论、
14	认知科学分支	人工智能	该实体类型描述的是模拟人类智能的计算机系统。包括机器学习、自然语言处理等技术。例如：专家系统、神经
15	认知科学分支	认知计算	该实体类型描述的是模拟人类认知过程的计算系统。结合认知科学和计算机科学技术。例如：IBM Watson系统、
16	认知科学分支	神经语言学	该实体类型描述的是研究语言加工的神经基础的交叉学科。采用脑成像等技术。例如：布洛卡区功能、失语症研
17	认知科学分支	心理语言学	该实体类型描述的是研究语言理解和产生的心理过程的学科。例如：词汇通达机制、句子加工模型等。
18	认知科学分支	社会语言学	该实体类型描述的是研究语言与社会因素关系的学科。例如：方言变异、语言态度、语码转换等。
19	理论与模型	符号处理模型	该实体类型描述的是将认知视为符号操作的计算模型。采用规则和表征系统。例如：物理符号系统假设、产生式
20	理论与模型	联结主义模型	该实体类型描述的是基于神经网络模拟认知过程的计算模型。强调并行分布式处理。例如：反向传播网络、Hopf
21	理论与模型	适应性表征	该实体类型描述的是认知系统为适应环境而形成的动态表征方式。例如：情境认知、具身认知等理论中的表征形
22	理论与模型	认知架构	该实体类型描述的是认知系统的整体组织和运行机制。例如：ACT-R架构、SOAR架构等计算认知模型。
23	理论与模型	知识图谱	该实体类型描述的是结构化表示领域知识的语义网络。包含实体、属性和关系。例如：Google知识图谱、领域本
24	理论与模型	事理图谱	该实体类型描述的是表示事件及其逻辑关系的知识图谱。强调时序和因果关系。例如：历史事件图谱、新闻事件
25	理论与模型	认知推理框架	该实体类型描述的是支持人类式推理的计算框架。结合逻辑和概率方法。例如：贝叶斯推理模型、案例推理系统
26	理论与模型	认知控制	该实体类型描述的是调控思维和行为的执行功能系统。包括抑制、转换和更新等成分。例如：斯特鲁普任务中的

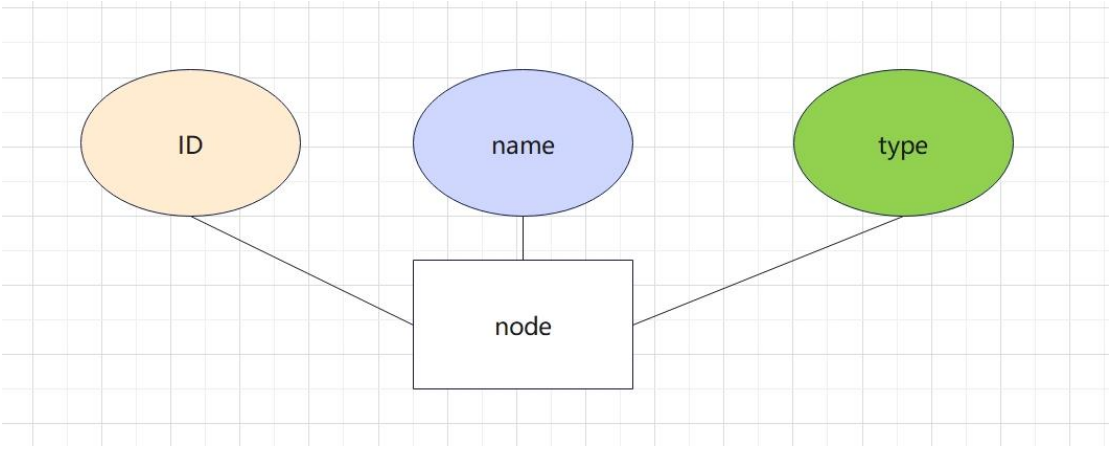
2.2 关系类型

以相同的方式解析出了主要的关系类型，并结合文本，进行了一定的解释，便于进行之后的实体，关系抽取。共计 60 种主要关系类型，具体表格如下：

序号	关系类型	关系	解释
1	基础认知关系	具有认知过程	该关系类型描述认知系统或模型包含的特定认知过程。主体是认知系统或模型，客体是认知过程。例如：'工作记忆系统'表示工作记忆包含信息暂时保持的过程。通过分析认知架构文档中的系统描述识别这种关系。文档2.1.5节详细描述了该
2	基础认知关系	属于认知系统	该关系是'具有认知过程'的反向关系，表示认知过程所属的认知系统或模型。主体是认知过程，客体是认知系统或模型。例如：'工作记忆系统'。文档2.1.1节对智能的基本概念进行了系统阐述，为理解认知系统提供了理论基础。
3	基础认知关系	处理信息类型	该关系类型描述认知实体对特定类型信息的处理能力。主体是认知实体，客体是信息类型。例如：'前额叶皮层'处理信息。文档9.5.1节详细探讨了认知概念与信息处理的关系，为理解这一关系提供了理论支持。
4	基础认知关系	被处理信息	该关系是'处理信息类型'的反向关系，表示信息被特定认知实体处理。主体是信息类型，客体是认知实体。例如：'语言区'。文档2.2.3节讨论了认知科学与其他知识单元的关联关系，为理解信息处理路径提供了参考。
5	基础认知关系	表现为模式	该关系类型描述认知实体展现的特定行为模式。主体是认知实体，客体是行为模式。例如：'人工认知系统'表现为模式。关于人工认知行为特征的描述详细解释了这一关系的理论基础和应用场景。
6	基础认知关系	属于行为主体	该关系是'表现为模式'的反向关系，表示行为模式所属的认知实体。主体是行为模式，客体是认知实体。例如：'问题解决'。文档1中对认知科学的定义为理解行为与认知主体的关系提供了概念框架。
7	基础认知关系	具有心理表征	该关系类型描述认知过程中的表征形式。主体是认知过程，客体是表征形式。例如：'语言理解'具有心理表征。语义网络计算的论述深入探讨了心理表征在认知过程中的作用机制。
8	基础认知关系	表征认知过程	该关系是'具有心理表征'的反向关系，表示表征形式对应的认知过程。主体是表征形式，客体是认知过程。例如：'概念形成'。文档3从认知神经科学视角的研究为理解表征与认知的关系提供了实证依据。
9	基础认知关系	涉及注意资源	该关系类型描述认知活动对注意资源的需求。主体是认知任务，客体是注意类型。例如：'多任务处理'涉及注意资源。对阅读的注意吸引阶段描述详细分析了注意资源的分配机制。
10	基础认知关系	需要记忆系统	该关系类型描述认知活动对记忆系统的依赖。主体是认知任务，客体是记忆系统。例如：'情景记忆提取'需要记忆系统。关于认知控制的论述为理解记忆系统的作用提供了理论支持。
11	理论与模型关系	提出理论	该关系类型描述研究者提出的认知理论。主体是研究者，客体是认知理论。例如：'布鲁纳'提出理论'认知学习理论'。对科学概念的发展历史，为理解理论发展脉络提供了重要参考。
12	理论与模型关系	由研究者提出	该关系是'提出理论'的反向关系。表示理论的提出者。主体是理论，客体是研究者。例如：'符号处理模型'由研究者提出。组内历史的梳理为追溯理论来源提供了系统依据。
13	理论与模型关系	扩展理论	该关系类型描述对已有理论的扩展和发展。主体是新理论，客体是基础理论。例如：'联结主义模型'扩展理论'神经网络'。对发展阶段的论述详细分析了理论演进的内在逻辑。
14	理论与模型关系	被理论扩展	该关系是'扩展理论'的反向关系，表示理论被哪些新理论扩展。主体是基础理论，客体是新理论。例如：'行为主义'被

2.3 节点属性

除了节点类型，我们为每个节点设计了多个属性，其中包括 ID（唯一表示一个节点），name（节点的通俗表示方法），type（节点属于的主要类型）。具体的节点表示及节点表如下：



	A	B	C	D	E
1	name: ID	type			
2	SUS得分	AdaptiveRepresentation			
3	丘脑后部脑	AdaptiveRepresentation			
4	个体认知	AdaptiveRepresentation			
5	主动悬架系	AdaptiveRepresentation			
6	互动性	AdaptiveRepresentation			
7	人工智能方	AdaptiveRepresentation			
8	人文学科达	AdaptiveRepresentation			
9	人类感知	AdaptiveRepresentation			
10	代价属性	AdaptiveRepresentation			
11	信息分析	AdaptiveRepresentation			
12	内容适应性	AdaptiveRepresentation			
13	内部表征	AdaptiveRepresentation			
14	分化性	AdaptiveRepresentation			
15	分析哲学方	AdaptiveRepresentation			
16	分级处理	AdaptiveRepresentation			
17	创造性思维	AdaptiveRepresentation			
18	剔除冗余平	AdaptiveRepresentation			
19	功能网络	AdaptiveRepresentation			
20	动态核形成	AdaptiveRepresentation			
21	动物意识	AdaptiveRepresentation			
22	双向传播	AdaptiveRepresentation			
23	发展历史	AdaptiveRepresentation			
24	可进化性	AdaptiveRepresentation			
25	哲学心理学	AdaptiveRepresentation			
26	因果关系	AdaptiveRepresentation			
27	在线视频交	AdaptiveRepresentation			
28	处理复杂性	AdaptiveRepresentation			
29	多模态感知	AdaptiveRepresentation			
30	大数据分析	AdaptiveRepresentation			
31	大脑发育	AdaptiveRepresentation			
32	奖励学习	AdaptiveRepresentation			
33	子符号表达	AdaptiveRepresentation			
34	学生自主学	AdaptiveRepresentation			
35	安全警报	AdaptiveRepresentation			
36	实验验证	AdaptiveRepresentation			
	< >	AdaptiveRepresentation	+		

第三章 抽取实体和关系

3.1 抽取方法选择

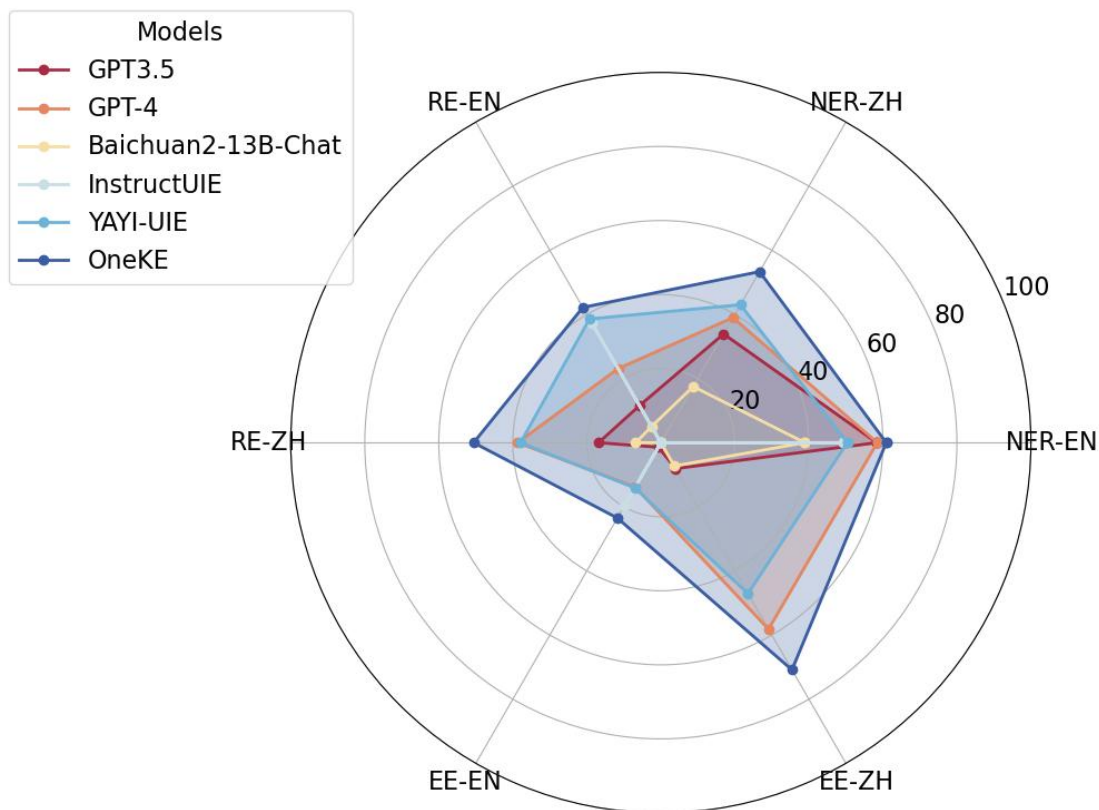
我们首先进行了多种尝试，包括 api 调用，deepke 抽取等等，发现这些方式都有各自的缺点，包括但不限于 api 限流不稳定，抽取出来效果不理想等等。经过我们多方面调研，最终选取了 OneKE 中英双语大模型知识抽取框架。



OneKE 是由蚂蚁集团和浙江大学联合研发的大模型知识抽取框架，具备中英文双语、多领域多任务的泛化知识抽取能力，并提供了完善的工具链支持。OneKE 以开源形式贡献给 OpenKG 开放知识图谱社区。

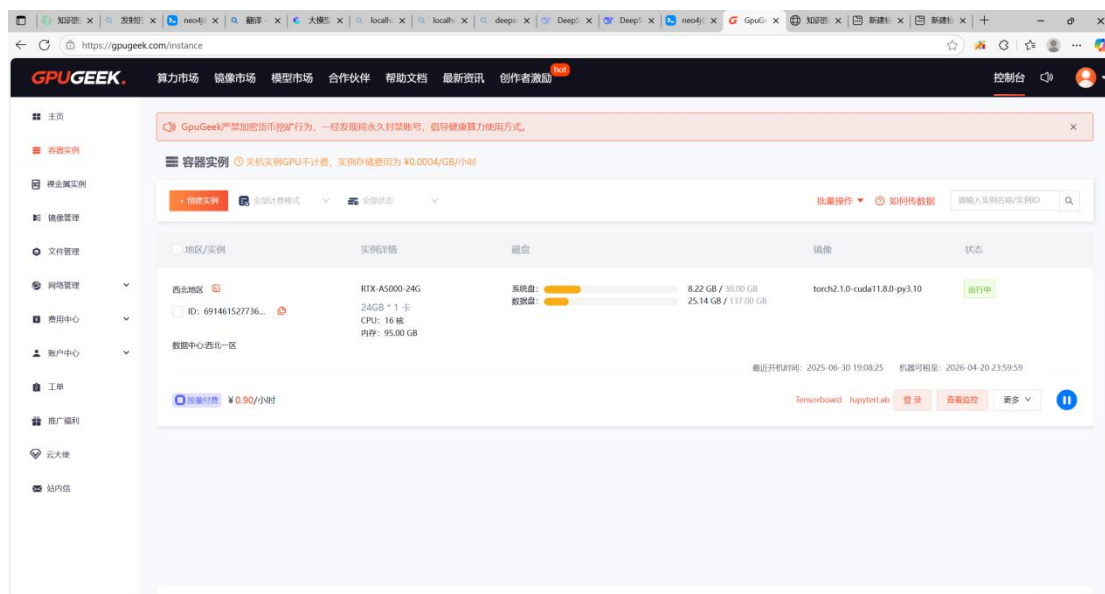
基于非结构化文档的知识构建一直是知识图谱大规模落地的关键难题之一，因为真实世界的信息高度碎片化、非结构化，大语言模型在处理信息抽取任务时仍因抽取内容与自然语言表述之间的巨大差异导致效果不佳，自然语言文本信息表达中因隐式、长距离上下文关联存在较多的歧义、多义、隐喻等，给知识抽取任务带来较大的挑战。针对上述问题，蚂蚁集团与浙江大学发布中英双语大模型知识抽取框架 OneKE，同时开源基于 Chinese-Alpaca-2-13B 全参数微调的版本。测评指标显示，OneKE 在多个全监督及零样本实体/关系/事件抽取任务上取得了相对较好的效果。

OneKE 在零样本泛化性上与其他大模型的对比结果：



3.2 OneKE 部署流程

训练和推理建议至少具备 20GB 的显存，所以需要租借一台服务器，这里使用的平台为 GpuGeek



镜像选择:

血与泪的教训告诉我们一定要提前选择对应的服务器镜像，这里我们选择

torch2.1.0-cuda11.8.0-py3.10

以匹配 oneke 框架

因为服务器上的文件系统原因，我们需要进入数据盘 gz-data，这样才能保证我们的存储空间足够，否则会导致磁盘容量不足运行下载模型失败。

```
(base) root@gz-ins-691461527736325:/gz-data/deepke# df -h
Filesystem                Size      Used Avail Use% Mounted on
overlay                    30G       8.5G   22G   29% /
tmpfs                      64M         0    64M    0% /dev
tmpfs                      378G         0   378G    0% /sys/fs/cgroup
shm                        95G         0    95G    0% /dev/shm
/dev/sda2                  439G      26G   395G    6% /usr/bin/nvidia-smi
JuiceFS:gpuzoom-user-data  1.0P      2.8T  1022T    1% /gz-fs
/dev/nvme0n1               137G      26G   112G   19% /gz-data
JuiceFS:gpuzoom-datasets   1.0P      96G   1.0P    1% /gz-datasets
JuiceFS:gpuzoom-models     1.0P     636G   1.0P    1% /gz-models
tmpfs                      378G      12K   378G    1% /proc/driver/nvidia
tmpfs                      76G       32M    76G    1% /run/nvidia-persistenced/socket
tmpfs                      378G         0   378G    0% /proc/asound
tmpfs                      378G         0   378G    0% /proc/acpi
tmpfs                      378G         0   378G    0% /proc/scsi
tmpfs                      378G         0   378G    0% /sys/firmware
tmpfs                      378G         0   378G    0% /sys/devices/virtual/powercap
```

如图 overlay 文件系统下只有 30G 空间

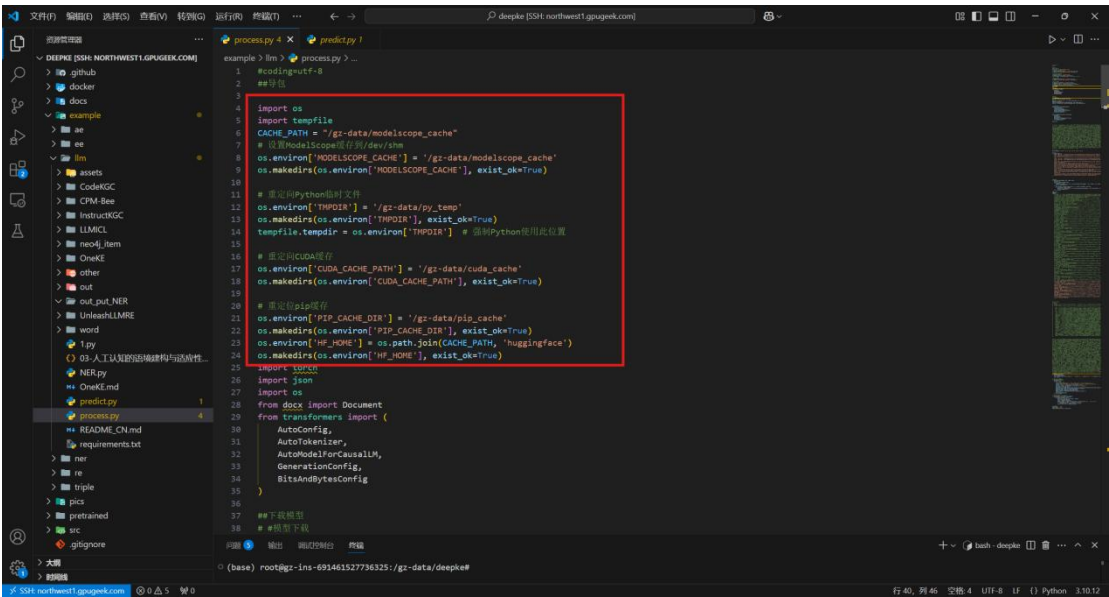
克隆源代码

clone <https://github.com/zjunlp/DeepKE.git>

进入对应目录

cd example/llm

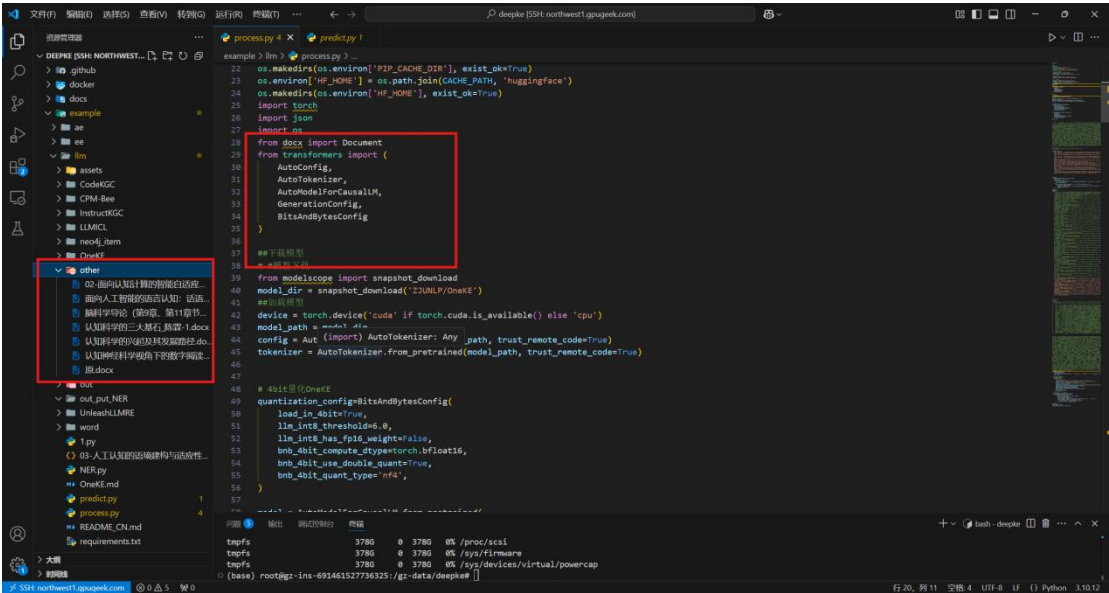
修改所示部分的代码



```
example > llm > process.py > ...
1 #coding=utf-8
2 ##方法
3
4 import os
5 import tempfile
6 CACHE_PATH = '/gz-data/modelscope_cache'
7 # 设置modelscope缓存到/dev/shm
8 os.environ['MODELSCOPE_CACHE'] = '/gz-data/modelscope_cache'
9 os.makedirs(os.environ['MODELSCOPE_CACHE'], exist_ok=True)
10
11 # 设置python临时文件
12 os.environ['TMPDIR'] = '/gz-data/py_temp'
13 os.makedirs(os.environ['TMPDIR'], exist_ok=True)
14 tempfile.tempdir = os.environ['TMPDIR'] # 强制python使用此位置
15
16 # 设置cuda缓存
17 os.environ['CUDA_CACHE_PATH'] = '/gz-data/cuda_cache'
18 os.makedirs(os.environ['CUDA_CACHE_PATH'], exist_ok=True)
19
20 # 设置pip缓存
21 os.environ['PIP_CACHE_DIR'] = '/gz-data/pip_cache'
22 os.makedirs(os.environ['PIP_CACHE_DIR'], exist_ok=True)
23 os.environ['HF_HOME'] = os.path.join(CACHE_PATH, 'huggingface')
24 os.makedirs(os.environ['HF_HOME'], exist_ok=True)
25
26 import json
27 import os
28 from docs import Document
29 from transformers import (
30     AutoConfig,
31     AutoTokenizer,
32     AutoModelForCausalLM,
33     GenerationConfig,
34     BitsAndBytesConfig
35 )
36
37 ##下载模型
38 # 模型下载
```


如图，因为代码逻辑自动会把模型下载到 overlay 文件系统下进行缓存，依旧会爆满，所以这些代码的目的是修改缓存目录和重定向模型加载位置。

下一步是把用于提取的文档放入 other 文件夹下

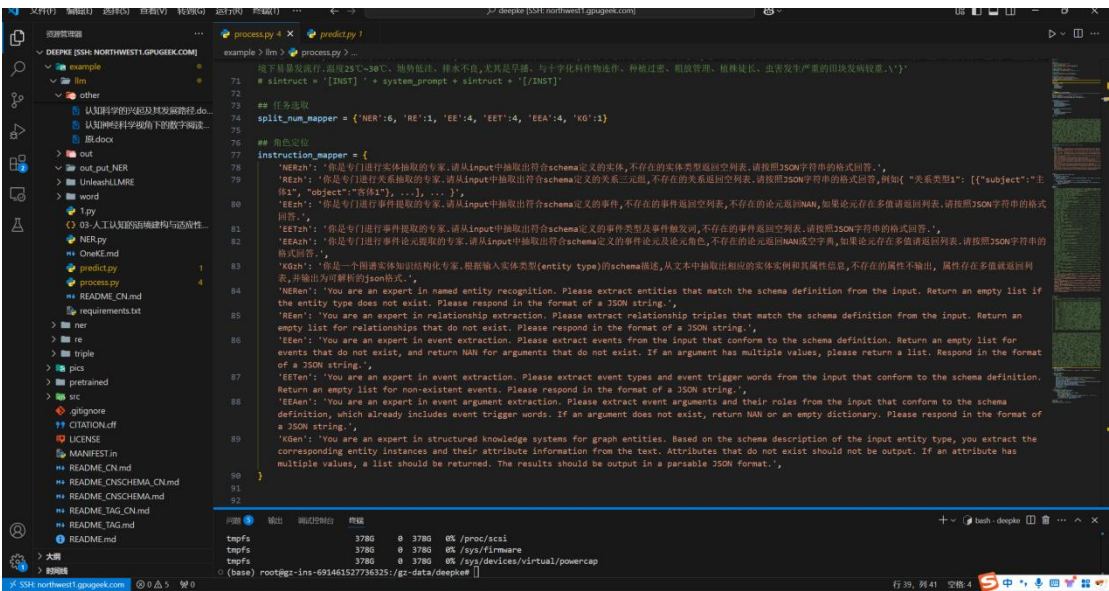


```
example > bin > process.py > ...
20 os.makedirs(os.environ['PIP_CACHE_DIR'], exist_ok=True)
21 os.environ['HF_HOME'] = os.path.join(CACHE_PATH, 'huggingface')
22 os.makedirs(os.environ['HF_HOME'], exist_ok=True)
23 import torch
24 import json
25 import os
26
27 from docx import Document
28 from transformers import (
29     AutoConfig,
30     AutoTokenizer,
31     AutoModelForCausalLM,
32     GenerationConfig,
33     BitsAndBytesConfig
34 )
35
36 ## 下载模型
37
38 from modelscope import snapshot_download
39 model_dir = snapshot_download('ZJUNLP/OnaKE')
40
41 ## 加载模型
42 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
43 model_path = os.path.join(model_dir, 'model')
44 config = AutoConfig.from_pretrained(model_path, trust_remote_code=True)
45 tokenizer = AutoTokenizer.from_pretrained(model_path, trust_remote_code=True)
46
47 ## 4bit量化OnaKE
48 quantization_config=BitsAndBytesConfig(
49     load_in_4bit=True,
50     llm_int8_threshold=6.0,
51     llm_int8_has_f16_weights=False,
52     bnb_4bit_compute_dtype=torch.float16,
53     bnb_4bit_use_double_quant=True,
54     bnb_4bit_quant_type='nf4',
55 )
56
57 model = AutoModelForCausalLM.from_pretrained(
58     model_path,
59     config=config,
60     quantization_config=quantization_config,
61     torch_dtype=torch.float16,
62     trust_remote_code=True,
63     device_map='auto',
64 )
65
66 model.eval()
67
68 # 生成文本
69 prompt = '你是一名专业的文档分类专家，请根据以下文档内容，对文档进行分类。'
70 input_ids = tokenizer(prompt).input_ids
71 with torch.no_grad():
72     output_ids = model.generate(input_ids, max_length=100)
73     output_ids = output_ids[input_ids.shape[0]:]
74     output_text = tokenizer.decode(output_ids)
75
76 # 输出结果
77 print(output_text)
```

这里需要下载 docx 和 modelscope 的库，用于提取文档和下载模型

`pip install pyhon-docx modelscope`

角色定位提示词：

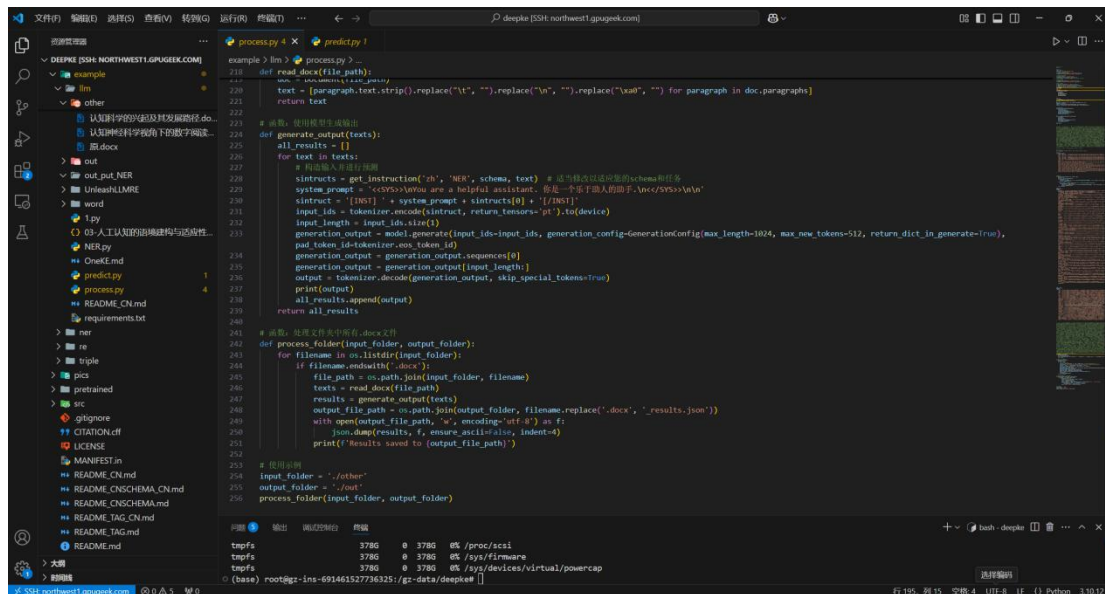


```
71 线下最易发生行、温度25℃~30℃、地势低洼、排水不良、尤其是旱稻、与十字花科作物连作、种植过密、粗放管理、植株矮小、虫害发生严重的田块发病较重。”)
72 # sinstruct = '[INST] ' + system_prompt + sinstruct + '[/INST]'
73
74 ## 任务选取
75 split_num_mapper = {'NER':6, 'RE':1, 'EE':4, 'EET':4, 'EEA':4, 'KO':1}
76
77 ## 角色定位
78 instruction_mapper = {
79     'NERzh': '你是专门进行实体抽取的专家，请从input中抽取符合schema定义的实体，不存在的实体类型返回空列表，请按照JSON字符串的格式回答。',
80     'REzh': '你是专门进行关系抽取的专家，请从input中抽取符合schema定义的关系三元组，不存在的关系返回空列表，请按照JSON字符串的格式回答，例如[{"subject": "主", "object": "客体", "...}], ... }',
81     'EETzh': '你是专门进行事件抽取的专家，请从input中抽取符合schema定义的事件，不存在的返回空列表，不存在的论元返回NaN，如果论元存在多个请返回列表，请按照JSON字符串的格式回答。',
82     'EEAzh': '你是专门进行事件论元抽取的专家，请从input中抽取符合schema定义的事件论元论元角色，不存在的论元返回NaN或空字典，如果论元存在多个请返回列表，请按照JSON字符串的格式回答。',
83     'K0zh': '你是一个精通实体知识结构化专家，根据输入实体类型(entity type)的schema描述，从文本中抽取相应的实体实例及其属性信息，不存在的属性不输出，属性存在多个值返回列表，并输出为可解析的json格式。',
84     'NERen': 'You are an expert in named entity recognition. Please extract entities that match the schema definition from the input. Return an empty list if the entity type does not exist. Please respond in the format of a JSON string.',
85     'REen': 'You are an expert in relationship extraction. Please extract relationship triples that match the schema definition from the input. Return an empty list for relationships that do not exist. Please respond in the format of a JSON string.',
86     'EETen': 'You are an expert in event extraction. Please extract events from the input that conform to the schema definition. Return an empty list for events that do not exist, and return NaN for arguments that do not exist. If an argument has multiple values, please return a list. Respond in the format of a JSON string.',
87     'EEAen': 'You are an expert in event argument extraction. Please extract event arguments and their roles from the input that conform to the schema definition, which already includes event trigger words. If an argument does not exist, return NaN or an empty dictionary. Please respond in the format of a JSON string.',
88     'K0en': 'You are an expert in structured knowledge systems for graph entities. Based on the schema description of the input entity type, you extract the corresponding entity instances and their attribute information from the text. Attributes that do not exist should not be output. If an attribute has multiple values, a list should be returned. The results should be output in a parsable JSON format.'
89 }
90
91
92
```

[illegible]

The screenshot shows a web browser window with the address bar displaying "deepke [SSH: NORTHWEST-1.GPUJEEK.COM]". The main content area displays a PDF document titled "process.py". The document is a research paper discussing cognitive models of knowledge representation and reasoning. It includes sections like "example", "task", "language", and "schema". The text discusses various types of knowledge representations such as "declarative knowledge" (陈述性知识), "procedural knowledge" (程序性知识), and "metacognitive knowledge" (元认知知识). It also mentions different levels of abstraction and the role of context in knowledge processing.

主体代码:



环境安装：

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

```
conda activate deepke-llm
```

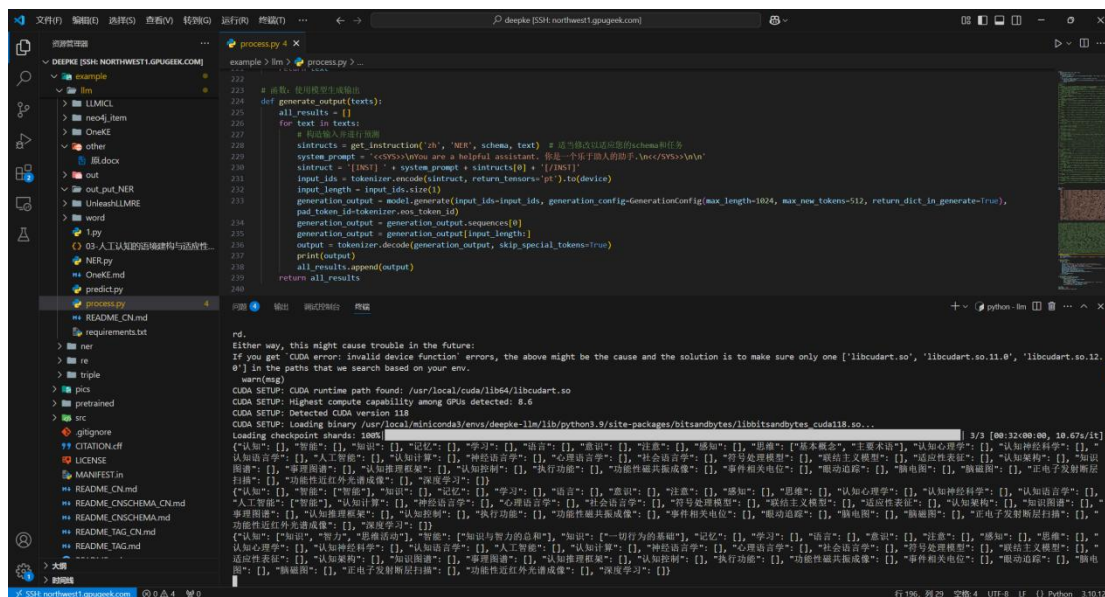
```
pip install -r requirements.txt
```

3.3 抽取实体和关系

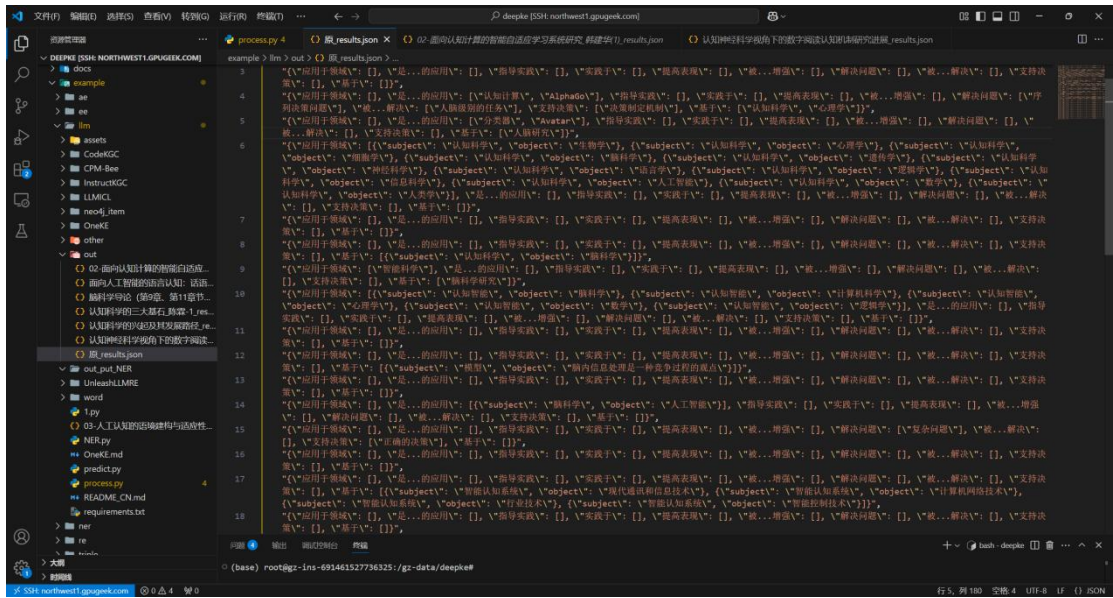
运行代码

```
python process.py
```

实体抽取过程：



关系抽取部分结果：



第四章 数据处理及完善

通过 OneKE 抽取出来的实体和关系以 json 文件的形式存储，存在几点问题

按照要求，Json 文件需要转换成 csv 文件

实体和关系的对应关系并不是非常准确，有些关系中的实体并没有在抽取出的实体当中，导致导入失败：

```
Available resources:
Total machine memory: 15.73GiB
Free machine memory: 2.771GiB
Max heap memory : 3.497GiB
Max worker threads: 16
Configured max memory: 1.277GiB
High parallel IO: true

WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe:putLong has been called by org.neo4j.internal.unsafe.UnsafeUtil (file:/D:/KnowledgeEngineering/lab4-neo4j/neo4j-community-5.26.4-windows/neo4j-community-5.26.4/lib/neo4j-unsafe-5.26.4.jar)
WARNING: Please consider reporting this to the maintainers of class org.neo4j.internal.unsafe.UnsafeUtil
WARNING: sun.misc.Unsafe:putLong will be removed in a future release

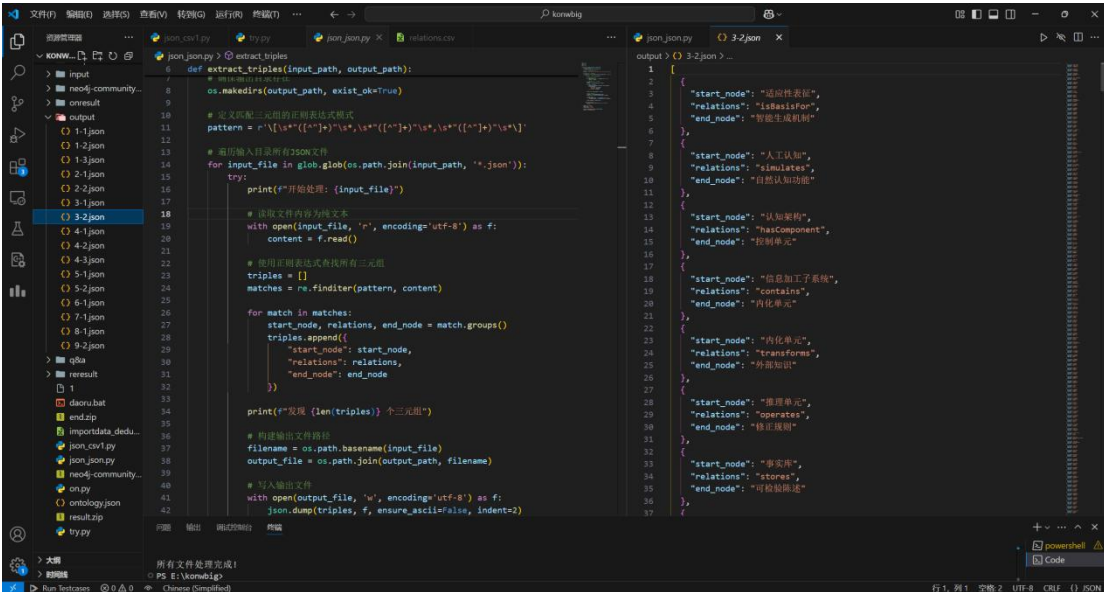
IMPORT FAILED in 0ms.
Data statistics is not available.
Peak memory usage: 0B
Database already exist. Re-run with '--overwrite-destination' to remove the database prior to import

WARNING Import failed. The store files in D:\KnowledgeEngineering\lab4-neo4j\neo4j-community-5.26.4-windows\neo4j-community-5.26.4\data\databases\neo4j are left as they are, although they are likely in an unusable state. Starting a database on these store files will likely fail or observe inconsistent records so start at your own risk or delete the store manually.

Error importing csv file.
Run with '--verbose' for a more detailed error message.
```

我们经过思考研究，提出并实现了全自动化处理数据及完善的策略：

1. 输出 json 到标准化 json 文件

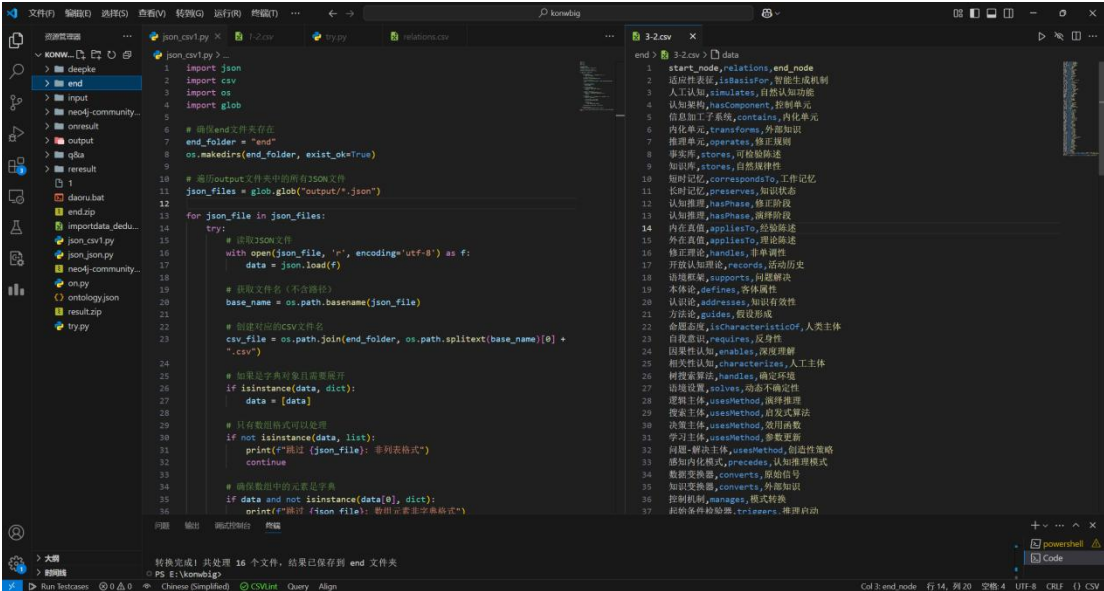


```
def extract_triples(input_path, output_path):
    os.makedirs(output_path, exist_ok=True)
    # 定义匹配三元组的正则表达式
    pattern = r'"start_node": "(.*)", "relations": "(.*)", "end_node": "(.*)"'
    # 遍历输入目录下的所有JSON文件
    for input_file in glob.glob(os.path.join(input_path, '*.json')):
        try:
            print(f"开始处理: {input_file}")
            # 读取文件内容为纯文本
            with open(input_file, 'r', encoding='utf-8') as f:
                content = f.read()
            # 使用正则表达式查找所有三元组
            triples = []
            matches = re.finditer(pattern, content)
            for match in matches:
                start_node, relations, end_node = match.groups()
                triples.append({
                    "start_node": start_node,
                    "relations": relations,
                    "end_node": end_node
                })
            print(f"发现 {len(triples)} 个三元组")
            # 构造输出文件路径
            filename = os.path.basename(input_file)
            output_file = os.path.join(output_path, filename)
            # 写入输出文件
            with open(output_file, 'w', encoding='utf-8') as f:
                json.dump(triples, f, ensure_ascii=False, indent=2)
```

```
output > 3-2.json > ...
1
[
  {
    "start_node": "适应性表征",
    "relations": "isBasisFor",
    "end_node": "智能生成机制"
  },
  {
    "start_node": "人工认知",
    "relations": "simulates",
    "end_node": "自然认知功能"
  },
  {
    "start_node": "认知架构",
    "relations": "hasComponent",
    "end_node": "控制单元"
  },
  {
    "start_node": "信息加工子系统",
    "relations": "contains",
    "end_node": "内化单元"
  },
  {
    "start_node": "推理单元",
    "relations": "operates",
    "end_node": "修正规则"
  },
  {
    "start_node": "事实库",
    "relations": "stores",
    "end_node": "可检验描述"
  }
]
```

利用正则表达式匹配，将收集到的不规则的 json 文件汇总为标准的形式，并去掉不满足要求的关系

2. 标准 json 文件到 csv 文件的转换



```
import json
import csv
import os
import glob

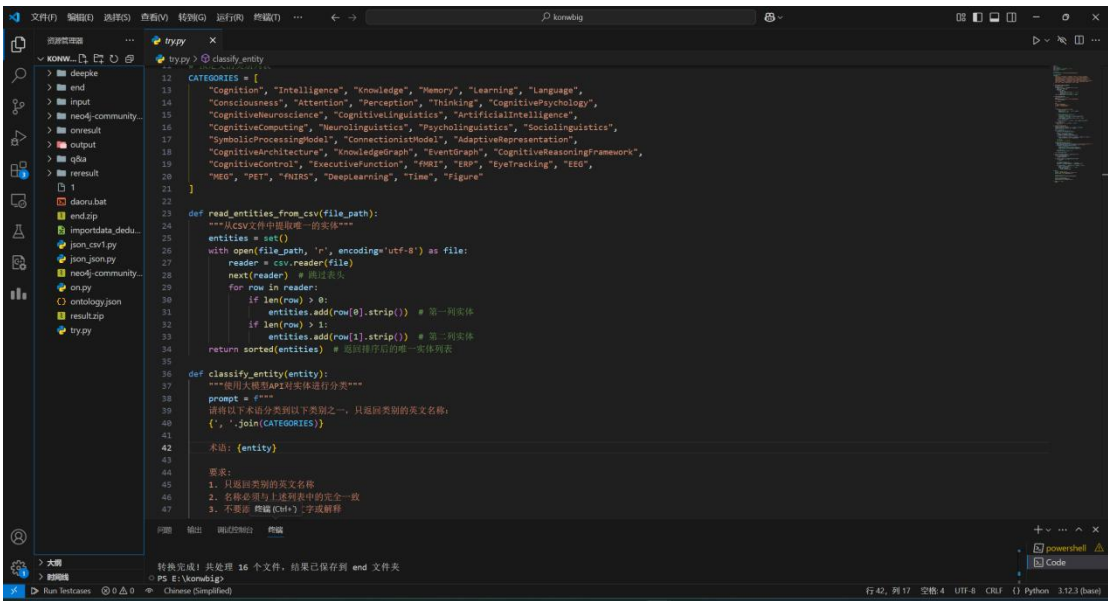
# 确保end文件夹存在
end_folder = "end"
os.makedirs(end_folder, exist_ok=True)

# 遍历output文件夹中的所有JSON文件
json_files = glob.glob("output/*.json")

for json_file in json_files:
    try:
        # 读取JSON文件
        with open(json_file, 'r', encoding='utf-8') as f:
            data = json.load(f)
        # 获取文件名（不含路径）
        base_name = os.path.basename(json_file)
        # 创建对应的CSV文件名
        csv_file = os.path.join(end_folder, os.path.splitext(base_name)[0] + ".csv")
        # 如果是字典列表，需要展开
        if isinstance(data, dict):
            data = [data]
        # 如果是列表，可以处理
        if not isinstance(data, list):
            print(f"跳过 {json_file}: 非列表格式")
            continue
        # 确保数据中的元素是字典
        if data and not isinstance(data[0], dict):
            print(f"跳过 {json_file}: 数据非字典格式")
            continue
```

```
end > 3-2.csv > data
1 start_node,relations,end_node
2 适应性表征,isBasisFor,智能生成机制
3 人工认知,simulates,自然认知功能
4 认知架构,hasComponent,控制单元
5 信息加工子系统,contains,内化单元
6 内化单元,transforms,外部知识
7 推理单元,operates,修正规则
8 事实库,stores,可检验描述
9 知识库,stores,自然规律性
10 短时记忆,correspondsTo,工作记忆
11 长时记忆,preserves,知识状态
12 认知推理,hasPhase,修正阶段
13 认知推理,hasPhase,保持阶段
14 内在真值,appliesTo,经验描述
15 外在真值,appliesTo,理论描述
16 修正理论,handles,非单调性
17 开闭认知理论,records,活动历史
18 语境框架,supports,问题解决
19 本体论,defines,客体属性
20 认识论,addresses,知识有效性
21 方法论,guides,数据形成
22 命题逻辑,hasCharacteristicOf,人类主体
23 自我意识,requires,反思性
24 因果性认知,enables,深度理解
25 相关性认知,characterizes,人工主体
26 树搜索算法,handles,确定环境
27 语境设置,solves,动态不确定性
28 逻辑主体,usesMethod,演绎推理
29 搜索主体,usesMethod,启发式算法
30 决策主体,usesMethod,效用函数
31 学习主体,usesMethod,参数更新
32 问题-解决主体,usesMethod,创造性策略
33 感知内化模式,precedes,认知推理模式
34 数据变换器,converts,数据信号
35 知识变换器,converts,外部知识
36 控制机制,manages,模式转换
37 起始条件,conditions,推理启动
```


3. 利用大模型 api 调用进行处理未在抽取实体中存在的部分



利用大模型进行智能的实体类别区分，保证了正确性

第五章 构建知识图谱三元组关系

5.1 三元组关系构建流程

1、首先，我们收集的数据格式为 word，PDF，每个文档的内容为与人工智能、认知科学等主题相关的文章。

2、然后我们通过在一oneKE 模型中输入设计好的 schema（节点类型和必要解释，关系类型和必要解释），进行实体和关系的抽取。抽取出来的文件为 json 形式。具体实现过程见第三章：抽取实体和关系。

3、接着我们编写 python 脚本，将抽取出来的 json 文件形式的实体和关系转化为三元组形式存储在 csv 中。

```
json_csv.py X
C: > Users > 34418 > Desktop > 文件资源 > data > json_csv.py
1 import json
2 import csv
3
4 # 读取JSON文件
5 with open('9-1.json', 'r', encoding='utf-8') as f:
6     data = json.load(f) # 支持JSON数组或对象
7
8 # 如果是字典对象且需要展开
9 if isinstance(data, dict):
10     data = [data]
11
12 # 写入CSV文件
13 with open('9-1.csv', 'w', encoding='utf-8', newline='') as f:
14     writer = csv.writer(f)
15
16     # 写入表头（取第一个对象的键）
17     writer.writerow(data[0].keys())
18
19     # 写入数据行
20     for row in data:
21         writer.writerow(row.values())
```

Python 代码实现了将 JSON 格式数据转换为 CSV 格式文件的功能,主要流程如下:

(1) 读取 JSON 数据:

第 4 行以 UTF-8 编码打开名为“9-1. json”的 JSON 文件

第 6 行使用 json.load() 方法加载文件内容到变量 data

(2) 数据格式统一化:

第 9-10 行智能判断: 若读取的数据是字典类型 (单个对象)

则将其转换为列表形式 (单个元素的列表), 确保后续处理一致性

(3) 写入 CSV 文件:

第 13 行以写入模式创建“9-1. csv”文件 (UTF-8 编码)

第 14 行创建 CSV 写入器对象

第 17 行自动提取第一个对象的键名作为 CSV 表头

第 20-21 行遍历所有数据对象, 将值逐行写入 CSV

(4) 技术亮点:

双 with 语句确保文件安全关闭

isinstance 动态判断数据类型

keys()/values() 自动提取表头与行数据

newline='' 参数避免 CSV 文件出现空行

通过这个代码,我们可以将下图的 json 数据格式转化为三元组关系并存储在 csv 表格中。

```
1 1
2 2 {"start_node": "注意吸引", "relations": "isTypeOf", "end_node": "认知加工过程"},
3 3 {"start_node": "识别聚焦", "relations": "isTypeOf", "end_node": "信息决策过程"},
4 4 {"start_node": "关联推理", "relations": "isTypeOf", "end_node": "语义理解过程"},
5 5 {"start_node": "学习建构", "relations": "isTypeOf", "end_node": "知识输出过程"},
6 6 {"start_node": "数字阅读", "relations": "hasPart", "end_node": "多模态感知"},
7 7 {"start_node": "注意吸引", "relations": "relatedTo", "end_node": "快速定向"},
8 8 {"start_node": "识别聚焦", "relations": "relatedTo", "end_node": "相关性判断"},
9 9 {"start_node": "关联推理", "relations": "relatedTo", "end_node": "意义通达"},
10 10 {"start_node": "学习建构", "relations": "relatedTo", "end_node": "思维重塑"},
11 11 {"start_node": "认知过程", "relations": "hasComponent", "end_node": "感知特征提取"},
12 12 {"start_node": "认知过程", "relations": "hasComponent", "end_node": "任务导向处理"},
13 13 {"start_node": "数字阅读", "relations": "influencedBy", "end_node": "认知需求"},
14 14 {"start_node": "学习建构", "relations": "promotes", "end_node": "问题解决"},
15 15 {"start_node": "内侧膝状体", "relations": "relatedTo", "end_node": "听觉处理"},
16 16 {"start_node": "初级视觉皮层", "relations": "relatedTo", "end_node": "视觉信息处理"},
17 17 {"start_node": "腹外侧前额叶皮层", "relations": "relatedTo", "end_node": "内容通路"},
18 18 {"start_node": "背外侧前额叶皮层", "relations": "relatedTo", "end_node": "空间通路"},
19 19 {"start_node": "左下顶叶皮层", "relations": "relatedTo", "end_node": "语义加工"},
20 20 {"start_node": "梭状回", "relations": "relatedTo", "end_node": "词形解析"},
21 21 {"start_node": "伏隔核", "relations": "relatedTo", "end_node": "奖赏反应"},
22 22 {"start_node": "前脑岛", "relations": "relatedTo", "end_node": "情绪过滤"},
23 23 {"start_node": "韦尼克区", "relations": "relatedTo", "end_node": "语言分辨"},
24 24 {"start_node": "角回", "relations": "relatedTo", "end_node": "跨通道联系"},
25 25 {"start_node": "左侧额下回", "relations": "relatedTo", "end_node": "记忆提取"},
26 26 {"start_node": "背内侧前额叶皮层", "relations": "relatedTo", "end_node": "语篇连贯"},
27 27 {"start_node": "左侧颞中回", "relations": "relatedTo", "end_node": "语义整合"},
28 28 {"start_node": "布罗卡区", "relations": "relatedTo", "end_node": "语言生成"},
```

转化后的表格如下：

	A	B	C	D
1	start_node	relations	end_node	
2	注意吸引	isTypeOf	认知加工过程	
3	识别聚焦	isTypeOf	信息决策过程	
4	关联推理	isTypeOf	语义理解过程	
5	学习建构	isTypeOf	知识输出过程	
6	数字阅读	hasPart	多模态感知	
7	注意吸引	relatedTo	快速定向	
8	识别聚焦	relatedTo	相关性判断	
9	关联推理	relatedTo	意义通达	
10	学习建构	relatedTo	思维重塑	
11	认知过程	hasComponent	感知特征提取	
12	认知过程	hasComponent	任务导向处理	
13	数字阅读	influencedBy	认知需求	
14	学习建构	promotes	问题解决	
15	内侧膝状体	relatedTo	听觉处理	
16	初级视觉皮层	relatedTo	视觉信息处理	
17	腹外侧前额	relatedTo	内容通路	
18	背外侧前额	relatedTo	空间通路	
19	左下顶叶皮	relatedTo	语义加工	
20	梭状回	relatedTo	词形解析	
21	伏隔核	relatedTo	奖赏反应	
22	前脑岛	relatedTo	情绪过滤	
23	韦尼克区	relatedTo	语言分辨	
24	角回	relatedTo	跨通道联系	
25	左侧额下回	relatedTo	记忆提取	
26	背内侧前额	relatedTo	语篇连贯	
27	左侧颞中回	relatedTo	语义整合	
28	布罗卡区	relatedTo	语言生成	
29	海马区	relatedTo	情景记忆	
30	听觉皮层	isPartOf	听觉系统	
31	视觉皮层	isPartOf	视觉系统	
32	腹侧通路	relatedTo	形状处理	
33	背侧通路	relatedTo	运动识别	
34	奖赏系统	relatedTo	选择倾向	
35	beta波	relatedTo	认知资源投入	
36	MMN	relatedTo	自动加工	

4、最后我们编写了自动化脚本，从之前抽取的关系 csv 文件中提取实体和本体，来完善 schema 定义的节点类型。使实体与关系更贴合，便于之后的 neo4j 导入数据。具体过程见第四章：数据处理及完善。

至此，我们得到了不同类型的实体所在 csv 表格，以及关系的 csv 表格，准备进行 noe4j 导入。

🔄 🖨️ > 文件资源 > import > entity

📁 📄 📄 📄 🗑️ ⬆️ 排序 🔍 查看 ⋮

名称	修改日期	类型	大小
📄 AdaptiveRepresentation.csv	2025/6/30 19:15	XLS 工作表	4 KB
📄 ArtificialIntelligence.csv	2025/6/30 19:15	XLS 工作表	4 KB
📄 Attention.csv	2025/6/30 19:15	XLS 工作表	6 KB
📄 Cognition.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 CognitiveArchitecture.csv	2025/6/30 19:15	XLS 工作表	4 KB
📄 CognitiveComputing.csv	2025/6/30 19:15	XLS 工作表	5 KB
📄 CognitiveControl.csv	2025/6/30 19:15	XLS 工作表	4 KB
📄 CognitiveLinguistics.csv	2025/6/30 19:15	XLS 工作表	5 KB
📄 CognitiveNeuroscience.csv	2025/6/30 19:15	XLS 工作表	5 KB
📄 CognitivePsychology.csv	2025/6/30 19:15	XLS 工作表	4 KB
📄 CognitiveReasoningFramework.csv	2025/6/30 19:15	XLS 工作表	6 KB
📄 ConnectionistModel.csv	2025/6/30 19:15	XLS 工作表	4 KB
📄 Consciousness.csv	2025/6/30 19:15	XLS 工作表	4 KB
📄 DeepLearning.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 EEG.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 ERP.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 EventGraph.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 ExecutiveFunction.csv	2025/6/30 19:15	XLS 工作表	4 KB
📄 EyeTracking.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 Figure.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 fMRI.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 fNIRS.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 Intelligence.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 Knowledge.csv	2025/6/30 19:15	XLS 工作表	3 KB
📄 KnowledgeGraph.csv	2025/6/30 19:15	XLS 工作表	4 KB

🔄 🖨️ > 文件资源 > import > relation

📁 📄 📄 📄 🗑️ ⬆️ 排序 🔍 查看 ⋮

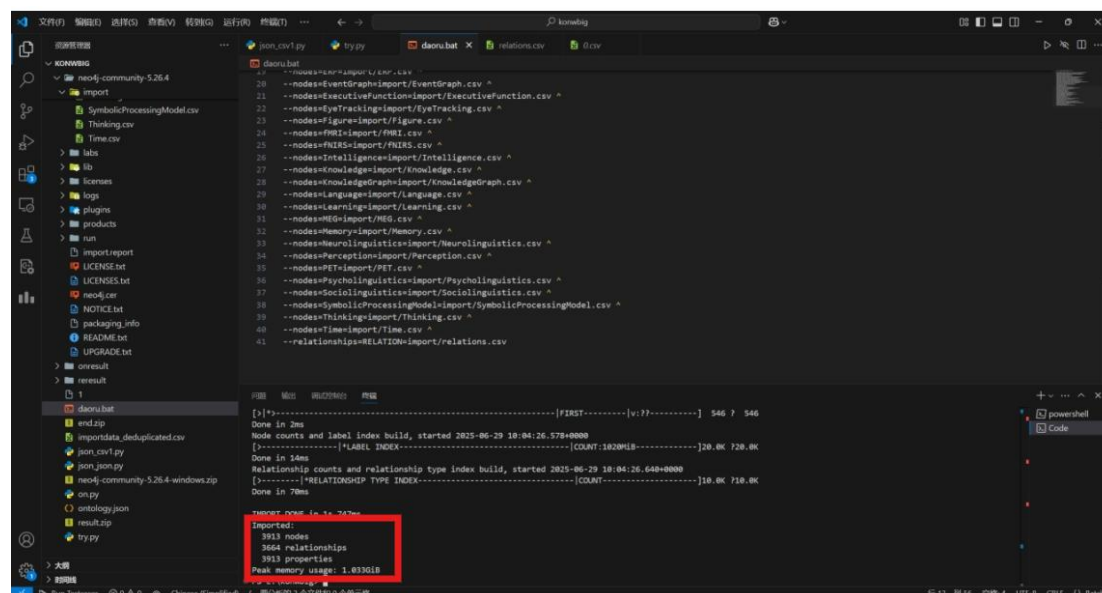
名称	修改日期	类型	大小
📄 relations.csv	2025/6/29 18:04	XLS 工作表	144 KB

第六章 neo4j 数据导入及可视化

6.1 批量导入

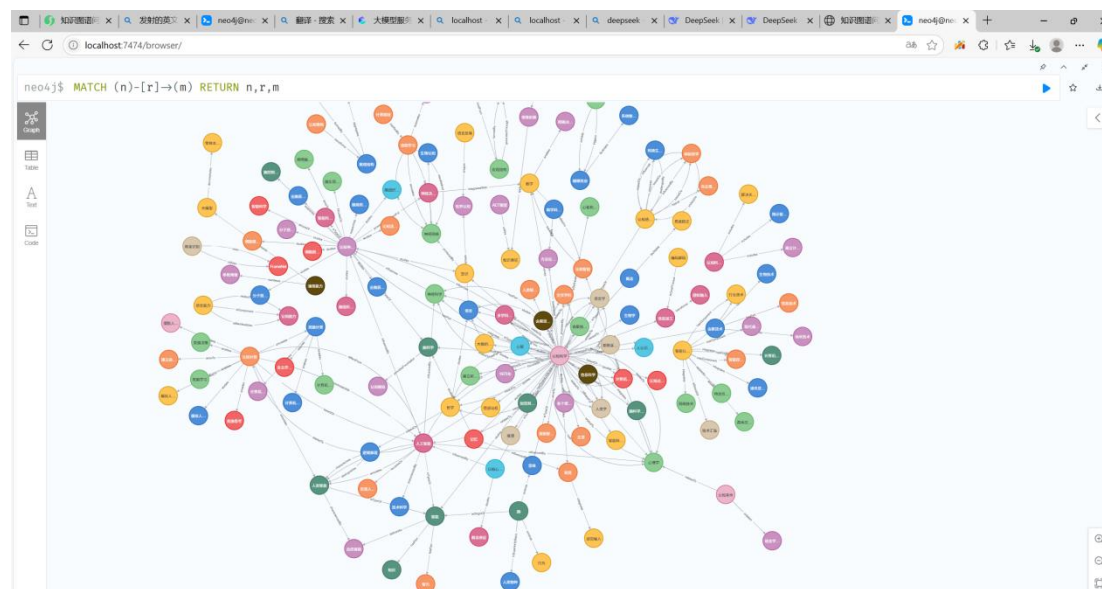
使用脚本批量导入，导入所有 csv 文件，导入过程见下图：

由图可知，一共导入了 3913 个节点，3664 条关系，3913 个实体。



6.2 noe4j 可视化

导入成功后，登录 localhost，我们可以看到已经构建好的知识图谱如下：



至此，说明构建图谱成功，并且可视化成功。且实体和关系数量较大，构建的图谱效果较好。

第七章 小结

本次知识图谱构建项目历经数据整合、Schema 设计、实体关系抽取、数据处理及可视化等关键阶段，在团队协作与技术探索中最终完成目标。

建立了包含 20 份文档、总规模达 30 万字的领域语料库。

定义 35 类实体节点及 60 类语义关系。评估包括 DeepKE、API 调用等方案后，选定 OneKE 中英双语大模型框架。面临并克服模型部署、GPU 资源调度（平台：GpuGeek，镜像：torch2.1.0-cuda11.8.0-py3.10）、环境配置等难点。

针对初始 JSON 数据存在的关系失配、格式不统一问题：开发正则匹配与批量转换脚本，实现 JSON 标准化。设计 Python 脚本，高效完成 JSON → CSV 三元组转换，实现数据处理流程自动化，显著提升可靠性与效率。

最终生成包含 3,913 个实体节点和 3,664 条关系边的知识图谱，且实体和关系数量较大构建的图谱效果较好。