

# 专业实习手册

学院\_\_\_\_\_ 数据科学与工程学院\_\_\_\_\_\_\_\_\_\_\_\_\_

系别\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

专业\_\_\_\_ \_\_\_\_\_数据科学与大数据技术\_\_\_\_\_\_\_

姓名\_\_\_\_\_\_温兆和\_\_\_\_\_\_\_学号\_\_10205501432\_\_\_

实习指导教师\_\_\_ 兰韵诗\_

实习单位\_\_\_华东师范大学数据科学与工程学院\_\_\_\_\_

实习日期 自\_\_\_2024\_\_\_年\_\_\_10\_\_\_月\_\_\_2\_\_\_日起

至\_\_\_2024\_\_\_年\_\_\_12\_ 月\_\_ 29\_ \_日止

实际实习\_\_\_\_\_\_\_\_\_\_\_13/65\_\_\_\_\_\_\_\_\_\_\_周天

目 录

1. 华东师范大学本科专业实习管理办法
2. 华东师范大学学生专业实习成绩评定表
3. 华东师范大学实习周报
4. 专业实习调查报告
5. 专业实习总结

**华东师范大学本科专业实习管理办法**

**一、专业实习的地位和作用**

专业实习是各专业实践教学的一个重要环节，其目的是理论联系实际，增强学生对社会、国情和专业背景的了解；使学生通过实习巩固和拓宽所学的理论知识，培养分析问题、解决问题的能力和创新能力；增强劳动观念，激励学生敬业、创业精神；积极探索实习教学与产学研一体化相结合的新途径，提高人才培养质量。

**二、专业实习的形式、内容和要求**

1．专业实习形式包括：（1）理工科的生产实习、野外实习、社会调查、毕业实习等；（2）文科的教学实习、社会调查、毕业实习等。

2．专业实习可根据实际情况采取集中实习、自主实习及其他多种方式。

3．专业实习内容由各专业根据培养目标及要求制定，并编制合适的实习计划。内容包括（1）实习的目的和意义；（2）实习的内容与要求；（3）实习的领导和指导工作；（4）实习纪律和有关注意事项；（5）成绩考核标准及办法；（6）实习的日程安排等。

**三、专业实习纪律**

为使专业实习顺利进行，培养学生良好学风，确保实习任务的圆满完成，特制定如下纪律：

1．遵守国家政策法令和《普通高等学校学生管理规定》、《高等学校学生行为准则》及我校的有关规章制度。

2．认真学习专业实习的有关文件和各项规定，明确专业实习的目的，端正态度，切实做好专业实习的各项工作。

3．服从院系和实习单位的领导，接受双方教师的指导。对实习工作如有意见和建议，应及时与院系指导教师沟通。

4．严格遵守实习单位的各项规章制度、自觉遵守纪律。实习期间，不迟到、不早退、不缺勤。因故请假，必须事先写出书面申请，经实习单位及我校指导教师同意。

5．积极参加各项业务实习活动，努力向指导教师学习，发扬勤奋好学、虚心求教、文明礼貌、艰苦奋斗的好作风。

6．认真写好实习日记、调查报告、实习总结等。

7．维护学校荣誉，树立大学生的高尚形象，防止一切不文明行为的发生。

8．爱护公物，在实习期间借用的物品必须按期归还，如有丢失损坏，必须照价赔偿。

9．团结友爱，关心集体，共同提高，克服生活和实习中遇到的各种困难。

**四、专业实习成绩的考核**

1．实习成绩考核是一项重要而严肃的工作，任何形式的专业实习都必须认真进行成绩考核与评定。

2．专业实习成绩评定采用A（优）、B（良）、C+（中）、C（及格）、F（不及格）记分。考核成绩不及格者不得毕业。

3．专业实习成绩评定包括两个方面：实习态度和实习效果。考核的要求是：

（1）实习态度（占20％）

实习生必须明确专业实习的目的和意义，积极主动地做好实习的各项工作。服从院系和实习单位的领导，尊重指导教师；严格要求自己，自觉遵守实习单位的有关规章制度；维护学校荣誉，树立大学生的高尚形象。对严重违反实习纪律、造成恶劣影响者，实习成绩作不及格处理。

（2）实习效果（占80％）

实习生应按照实习大纲的要求认真完成实习规定的各项任务。记录实习情况、心得体会等；对组织的教学参观、专题报告都要记录并加以整理；在完成实习任务的同时，结合实际，自选专题进行社会调查，写出报告（可作为实习报告内容之一）；或能独立完成教师布置的专题作业或对某些问题有独到见解及合理化建议；实习结束前写好实习报告，对政治思想和业务收获进行全面总结，实习报告质量必须达到实习大纲的要求。

4．各专业可根据本专业实际情况，按照上述考核要求制定具体细致的评分标准。

5．专业实习成绩由实习单位指导教师初评，写出评语和评分，最后由各专业实习领导小组审定。

**华东师范大学学生专业实习成绩评定表**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **姓名** | | **温兆和** | **性别** | | **男** | **专业年级** | **2021** |
| **实习单位** | | **华东师范大学数据科学与工程学院** | | | | **实习时间** | **2024.10-2024.12** |
| **实习单位评分及评语** | **实习态度（20分）** | **评分** | **20** | | | | |
| **评语** | **该同学实习态度端正，保证每周组会正常出勤，并主动积极学习研究课题，完成交付的研究工作。** | | | | |
| **实习效果（80分）** | **实习完成质量**  **（30分）** | **评分** | **30** | | | |
| **评语** | **该同学实习期间认真完成交付的工作，交付工作质量达标。** | | | |
| **实习**  **工作量**  **（10分）** | **评分** | **10** | | | |
| **评语** | **该同学实习期间对语法纠错任务和问题构建任务进行了系统地了解和调研，并完成多个实验，工作量达标。** | | | |
| **实习**  **报告**  **（作业）（20分）** | **评分** | **20** | | | |
| **评语** | **该实习报告能够总结每周的进展和收获，从浅入深，有层次，阐述较全面清晰。** | | | |
| **实习**  **总结**  **（20分）** | **评分** | **18** | | | |
| **评语** | **该实体总结能够归纳实习的收获并有所体会，为之后的研究工作打下了较好的基础。** | | | |
| **总分= 98 ，折合五级制：A** | | | | | |
| **指导教师（签名）\_\_\_\_\_\_\_\_\_\_ 实习单位（签章）\_\_\_\_\_\_\_\_\_\_**  **年 月 日** | | | | | |
| **学校意见** | **学校实习带队老师：\_\_\_\_\_\_\_\_\_\_ 系签章\_\_\_\_\_\_\_\_\_\_**  **年 月 日** | | | | | | |

**华东师范大学实习生日记**

2024.10.01

在本次任务中，我需要将所有文本文件输入大语言模型，让LLM判断内容属于“技术解析”还是“客户案例”。

具体来说，我用ollama调用qwen2.5:7b-instruct大语言模型，输入文本并指导大模型给出符合要求的回答，并根据大模型的回答判断相应内容所属类别。

import ollama

import torch

def qwen(prompt):

response = ollama.generate(model='qwen2.5:7b-instruct', prompt=prompt)

return response['response']

def categorize\_content(content):

prompt = (

"你是一个专业的文章分类专家。请问下面这篇文章的类型是技术解析吗？如果是，请回复“技术解析”；如果不是，请回复“客户案例”：\n\n"

"内容：\n"

f"{content}\n\n"

"请回复如下格式：\n"

"{'answer': '客户案例' 或 '技术解析'}"

)

answer = qwen(prompt)

# 解析模型返回的答案

if "客户案例" in answer:

return "客户案例"

elif "技术解析" in answer:

return "技术解析"

else:

return "false!"

2024.12.28

这一次的实习任务是分别实现RAW LLM、DPR和bm25算法在msqa和procqa测试集上的问答及评估。具体来说，就是先直接用qwen2.5:7b-instruct-fp16大语言模型对测试集中的每个问题生成答案，再分别用BM25和DPR算法从知识库中检索出与相应问题相关的文章，放到prompt中输入进大模型，并生成相应问题的回答。

使用qwen2.5:7b-instruct-fp16模型生成答案：

def qwen\_llm(prompt):

response = ollama.generate(model='qwen2.5:7b-instruct-fp16', prompt=prompt)

response\_str = extract\_answer(response['response'])

return response\_str

生成prompt：

def generate\_prompt(question, knowledge\_base\_context):

prompt = f"""

# Role

You are a proficient expert specializing in answering questions about the Lisp programming language.

### System Instructions:

1. Analyze the question carefully to understand the user’s intent.

2. \*\*If previous\_relevant\_qa or knowledge\_base\_context is provided\*\*:

- If \*previous\_relevant\_qa\* is highly similar to the \*Given Question\*, directly use the answer from previous\_relevant\_qa without modifications.

- If \*previous\_relevant\_qa\* is not available or not highly similar, use the information from the knowledge\_base\_context to provide a well-informed answer.

- If neither context provides sufficient information to answer the question, respond with: "Unable to answer based on the available knowledge."

- Ensure responses are accurate, relevant, and avoid adding unrelated information.

3. \*\*If no context is provided\*\*:

- Use your in-depth knowledge of the Lisp programming language to answer the question directly.

- If the LLM's internal knowledge base does not provide a sufficient answer, respond with: "Unable to answer based on the available knowledge."

Context:

- Knowledge Base: {knowledge\_base\_context}

- Previous Relevant Q&A:

Given Question: {question}

Respond to the given question in JSON format, structured as "answer": "your answer"

"""

return prompt

直接用大模型生成答案：

knowledge\_base\_context\_raw\_llm = ['' for \_ in test\_questions]

raw\_llm\_generated\_answers, raw\_llm\_bertscore\_f1\_list, raw\_llm\_cosine\_sim\_score\_list, raw\_llm\_bleu\_score\_list, raw\_llm\_meteor\_score\_list, raw\_llm\_rouge1\_list, raw\_llm\_rouge2\_list, raw\_llm\_rougeL\_list = test(test\_questions, test\_answers, knowledge\_base\_context\_raw\_llm)

先用BM25算法在知识库中对问题进行检索，再将检索出的内容一并输入大模型生成答案：

# 文本编码

def preprocess(text):

text = text.lower()

text = text.translate(str.maketrans('', '', string.punctuation))

return word\_tokenize(text)

tokenized\_corpus = [preprocess(doc) for doc in knowledge\_base]

bm25 = BM25Okapi(tokenized\_corpus)

# 遍历每个问题，计算 BM25 得分并返回最相关的 6 个文档

def get\_top\_bm25\_answers(question, top\_n=6):

tokenized\_query = preprocess(question)

scores = bm25.get\_scores(tokenized\_query)

top\_indices = sorted(range(len(scores)), key=lambda i: scores[i], reverse=True)[:top\_n]

return [knowledge\_base[i] for i in top\_indices]

knowledge\_base\_context\_bm25 = []

for question in test\_questions:

top\_answers = get\_top\_bm25\_answers(question)

combined\_context = " ".join(top\_answers)

knowledge\_base\_context\_bm25.append(combined\_context)

bm25\_generated\_answers, bm25\_bertscore\_f1\_list, bm25\_cosine\_sim\_score\_list, bm25\_bleu\_score\_list, bm25\_meteor\_score\_list, bm25\_rouge1\_list, bm25\_rouge2\_list, bm25\_rougeL\_list = test(test\_questions, test\_answers, knowledge\_base\_context\_bm25)

先用DPR算法在知识库中对问题进行检索，再将检索出的内容一并输入大模型生成答案：

# 加载模型和tokenizer

question\_encoder = DPRQuestionEncoder.from\_pretrained('facebook/dpr-question\_encoder-single-nq-base')

context\_encoder = DPRContextEncoder.from\_pretrained('facebook/dpr-ctx\_encoder-single-nq-base')

question\_tokenizer = DPRQuestionEncoderTokenizer.from\_pretrained('facebook/dpr-question\_encoder-single-nq-base')

context\_tokenizer = DPRContextEncoderTokenizer.from\_pretrained('facebook/dpr-ctx\_encoder-single-nq-base')

# 将模型移动到 GPU

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

question\_encoder.to(device)

context\_encoder.to(device)

# 对知识库中的每个文档进行编码（生成向量）

def encode\_documents(documents, batch\_size, device):

encoded\_documents = []

for i in range(0, len(documents), batch\_size):

batch\_docs = documents[i:i + batch\_size]

inputs = context\_tokenizer(batch\_docs, return\_tensors="pt", truncation=True, padding=True, max\_length=512)

# 将所有的输入张量移到 GPU 上

inputs = {key: value.to(device) for key, value in inputs.items()}

with torch.no\_grad():

doc\_embeddings = context\_encoder(\*\*inputs).pooler\_output

encoded\_documents.append(doc\_embeddings)

return torch.cat(encoded\_documents, dim=0)

# 编码问题

def encode\_question(question, device):

inputs = question\_tokenizer(question, return\_tensors="pt", truncation=True, padding=True, max\_length=512)

# 将问题的输入张量移到 GPU 上

inputs = {key: value.to(device) for key, value in inputs.items()}

with torch.no\_grad():

question\_embedding = question\_encoder(\*\*inputs).pooler\_output

return question\_embedding

# 编码知识库文档

encoded\_docs = encode\_documents(knowledge\_base, 32, device)

knowledge\_base\_context\_dpr = []

# 对每个问题进行检索

for question in test\_questions:

question\_embedding = encode\_question(question, device)

similarities = cosine\_similarity(question\_embedding, encoded\_docs)

top\_k = torch.topk(similarities, k=6)

top\_answers = []

for idx in top\_k.indices:

top\_answers.append(knowledge\_base[idx]) # 将文档内容追加到列表中

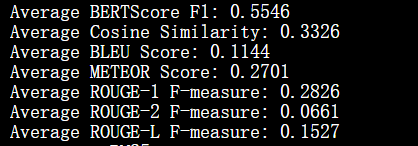
combined\_context = " ".join(top\_answers)

knowledge\_base\_context\_dpr.append(combined\_context)

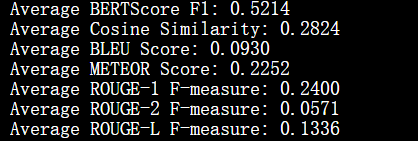
dpr\_generated\_answers, dpr\_bertscore\_f1\_list, dpr\_cosine\_sim\_score\_list, dpr\_bleu\_score\_list, dpr\_meteor\_score\_list, dpr\_rouge1\_list, dpr\_rouge2\_list, dpr\_rougeL\_list = test(test\_questions, test\_answers, knowledge\_base\_context\_dpr)

生成的回答用bertscore\_f1、cosine\_sim\_score、bleu\_score、meteor\_score和rouge（包括rouge1、rouge2和rougeL）评估。各个算法的指标如下：

RAW LLM：



BM25：



DPR：

