Embedding 微调

结果

原数据

模型\指标	precise@1	precise@2	precise@5
Openai emb-ada 002	42.7%	51.2%	65.3%
Openai emb-3-small	43.5%	50.8%	65.8%
Openai emb-3-large	47%	59.4%	73.5%
acge	51.7%	62.8%	79%
Acge 微调	53.8%	66.2%	84.6%
Bge	50.8%	61.5%	74.3%
Bge 微调	51.7%	66.2%	**85.4%**

新数据

模型\指标	precise@1	precise@2	precise@5
Acge 微调	50.6%	68.7%	84.8%
Bge 微调	51.2%	67.1%	81.5%

【注意】

初步测试使用 query_instruction_for_retrieval 效果没有不使用好，可能是数据量太小的原因

precise@n 计算方式

对于每一个 query，会在数据库中召回 5 个最相似的结果

例如

对于 query:可是健康告知中，还是要求甲状腺穿刺检测为良性才行啊，这个我明显不符合要求

召回的结果：pred = ["甲状腺结节","胆管炎","胰腺囊肿","宫颈上皮内瘤变","甲状腺癌"]

真实标签：label = ["甲状腺结节", "甲状腺癌"]

- precise@1：["甲状腺结节", "甲状腺癌"] 不是 pred[:1]的子集，预测不对
- precise@2：["甲状腺结节", "甲状腺癌"] 不是 pred[:2]的子集，预测不对
- precise@5：["甲状腺结节", "甲状腺癌"] 是 pred[:5]的子集，预测正确

依据

https://github.com/FlagOpen/FlagEmbedding/tree/master

微调步骤

数据集

- 原始数据

disease.xlsx 一共 4022 条

  - Question:问题
  - Thinking:思考过程，这里没有用到
  - Label:应该检索出的正确答案

	Question	Thinking	Label1	Label2	Label3	Label4	Label5
0	你好，体检有肺结节，有啥可以买的长期医疗保险，三甲医院医生说没问题，每年复查就行	客户提到的健康问题是体检发现肺结节，且医生建议每年复查，表明肺结节是当前主要关注的健康状况，...	肺结节	NaN	NaN	NaN	NaN
1	具体我没看到他的报告，问他是这么说的，有点脂肪肝	客户提到的症状是"有点脂肪肝"，这是直接从客户描述中得出的信息，因此不需要进一步推理。在提供...	脂肪肝	NaN	NaN	NaN	NaN

- 标签

疾病类型一共有 211 个【去重】

数据集预处理【特定处理】

对于 disease.xlsx 中的数据

- 删除 Question 为空的行
- 删除 Label 都为空的行
- 对每一行增加负样本标签
- 训练集和测试集的划分

数据集划分

- 训练集 jsonl

数据类型：{'query':'xxx','pos':'xxx','neg':'xxx'}

数据量：2103 条

- 测试集 jsonl

数据类型：{'query':'xxx','pos':'xxx'}

数据量：234 条

挖掘 hard negatives

sh run_hard_negatives.sh

```
python -m FlagEmbedding.baai_general_embedding.finetune.hn_mine \
--model_name_or_path bge-large-zh-v1.5 \
--input_file ./embedding_finetune/data_process/finetune_data.jsonl \
--output_file ./embedding_finetune/data_process/finetune_data_minedHN.jsonl \
--range_for_sampling 2-200 \
--negative_number 5 \
--use_gpu_for_searching
```

- model_name_or_path:原始模型路径，对应于没有微调的模型
- input_file:上述数据集划分时的训练集
- output_file:经过挖掘硬负样本后的训练集
- negative_number:给每个 Question 挖掘的硬负样本数量

训练微调

sh finetune.sh

```
CUDA_VISIBLE_DEVICES=3,5,6,7 torchrun --nproc_per_node 4 \
-m FlagEmbedding.baai_general_embedding.finetune.run \
--output_dir output_noinstruct \
--model_name_or_path bge-large-zh-v1.5 \
--train_data ./embedding_finetune/data_process/finetune_data_minedHN.jsonl \
--learning_rate 1e-5 \
--fp16 \
--num_train_epochs 10 \
--per_device_train_batch_size 1 \
--dataloader_drop_last True \
--normlized True \
--temperature 0.02 \
--query_max_len 64 \
--passage_max_len 256 \
--train_group_size 2 \
--negatives_cross_device \
--logging_steps 10 \
--save_steps 1000 \
--query_instruction_for_retrieval ""
```

- output_dir:微调后模型参数的存放路径
- model_name_or_path:原模型参数路径
- train_data:微调数据集，是前面经过挖掘 hard negatives 的数据集
- query_instruction_for_retrieval:提示词【如果挖掘 hard negatives 时使用了这个参数，微调时应该也使用】

评估

建立数据索引【create_index.py】

embeddings 模型的初始化

```
Python
#AzureOpenAIEmbeddings
embeddings = AzureOpenAIEmbeddings(
    deployment="text-embedding-ada-002",
    openai_api_key='b2753222b7ae4110bb3ef10835085c3e',
    azure_endpoint='https://insnailgpt4us.openai.azure.com/',
    openai_api_version='2024-02-01'
)
embeddings = AzureOpenAIEmbeddings(
    deployment="text-embedding-3-small",
    openai_api_key='b2753222b7ae4110bb3ef10835085c3e',
    azure_endpoint='https://insnailgpt4us.openai.azure.com/',
    openai_api_version='2024-02-01'
)
embeddings = AzureOpenAIEmbeddings(
    deployment="text-embedding-3-large",
    openai_api_key='b2753222b7ae4110bb3ef10835085c3e',
    azure_endpoint='https://insnailgpt4us.openai.azure.com/',
    openai_api_version='2024-02-01'
)

# bge
model_path = '../../bge-large-zh-v1.5'
query_instruction_for_retrieval = ''
embeddings =
BGEEmbedding(model_path,query_instruction_for_retrieval)
embeddings =
BGEEmbedding_finetune(model_path,query_instruction_for_retrieval)

#acge
model_path = '../../acge_text_embedding'
embeddings = ACGE(model_path)
embeddings = ACGE_finetune(model_path)
```

建立 index

```
Python
input_file = '../data/label.json'
index_name = 'acge_test_fine1' #对于一个新的 embeddings 模型，应该取一个新的名字
corpus = get_corpus_list(input_file)
store_to_db(corpus,index_name,embeddings)
```

准确率评估【evaluate_precise.py】

计算准确率

```
Python
index_name = 'acge_test_fine1'
model_path = '../../bge-large-zh-v1.5'
query_instruction_for_retrieval = ''
embeddings =
BGEEmbedding(model_path,query_instruction_for_retrieval)
db = get_es_db(index_name,embeddings)

query_file = '../data/query.jsonl'
for line in tqdm(open(query_file)):
    query_dic = json.loads(line.strip())
    query = query_dic['query']
    retrieval_result = search(query,db,k=5)
    ground_truth = query_dic['pos']

    retrieval_result_list.append(retrieval_result)
    ground_truth_list.append(ground_truth)

metrics = evaluate(preds=retrieval_result_list,
labels=ground_truth_list, cutoffs=[1,2,5])
print('准确率：',metrics)
```