知识库构建目标规划

目标一

1.熟悉开发环境和开发流程

1.环境搭建

2.资源库配置

2.调研embedding模型内存占用情况测评

（1）文档切分中的Chunk和Overlap

在处理较长文档或文本时，将其分割成若干小块，每块称为一个Chunk。在这个特定的切分策略中，每个Chunk由最多N个Token组成。Token通常指文本中的单词或符号，取决于具体语境。而Overlap是相邻Chunk之间共有的Token数。举个例子：

> ```
> 1 每Chunk 200 Token，Overlap 20。在这个例子中，每个Chunk由最多200个Token组成。
> Overlap为20，意味着相邻的Chunks会有20个Token是重复的，从而确保文本的连贯性。例如，如果
> 某个文本段落共有230个Token，它将被分成两个Chunks：第一个Chunk将有200个Token，第二个
> Chunk将有30个Token（因为230-200=30），并且这两个Chunks之间将有20个Token重叠。
> ```

这种切分方式有助于确保在将长文档送入诸如大型语言模型进行Embedding或处理时，能够保持文本的语义连贯性，同时又能满足模型处理长度有限的输入的要求。切分文档时考虑Chunk的大小和Overlap的数量，对于提高模型处理效率和文本的语义理解都是十分重要的。

（2）模型调研

与大模型类似，Embedding也是使用模型来实现的，只不过Embedding模型更为轻量。一般都在2G以内。

RAG emmbedding模型选取原则

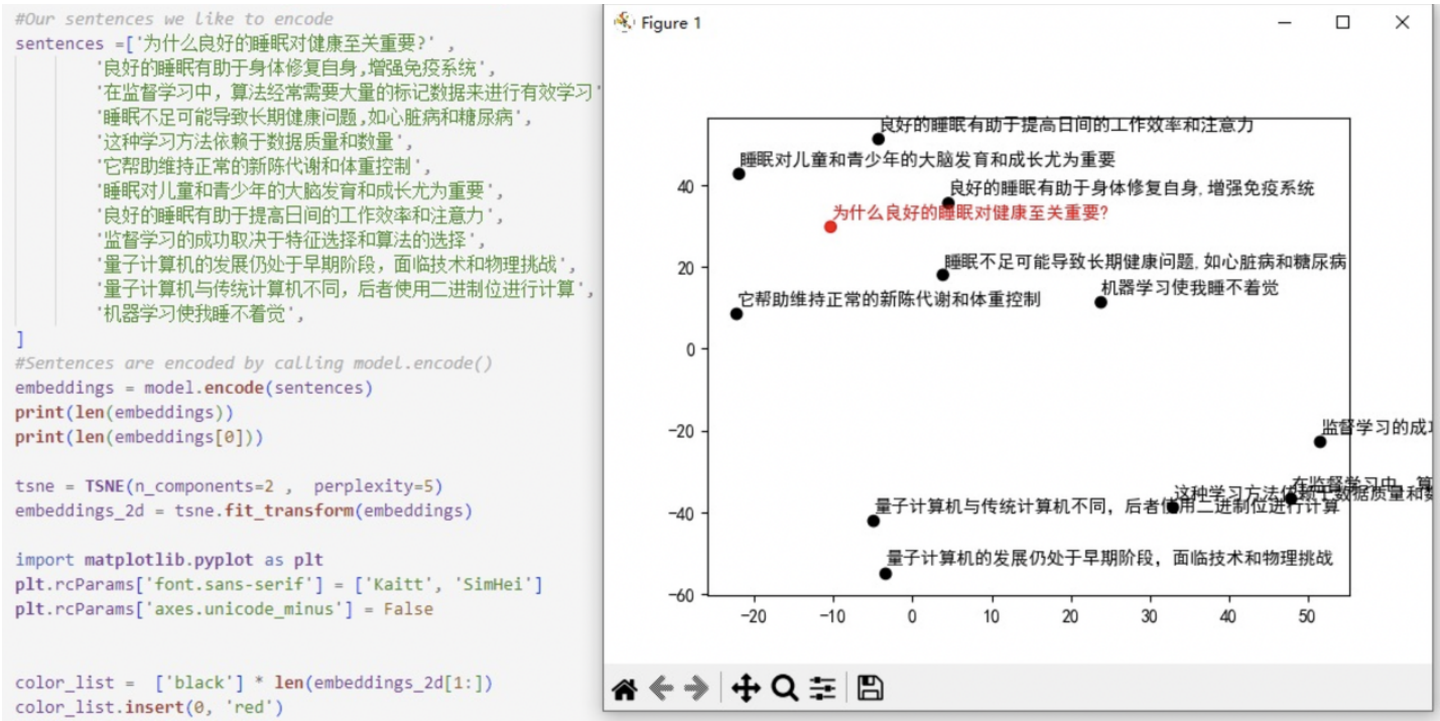
①**序列长度**取决于query所对应的答案和token的长度，选择与之相对应的。

②**embedding维度**并不是越大越好，取决于语料的语义是否丰富。

③**模型大小**取决于自身设备。

④ 用一个**小的任务**做可视化,初步观察,可以作为参考并不绝对。

```python
###任务可视化代码
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('emmbedding模型路径')
sentences = [
        '为什么良好的睡眠对健康至关重要?' ,
        '良好的睡眠有助于身体修复自身,增强免疫系统'……
]
embeddings = model.encode(sentences)

tsne = TSNE(n_components=2 ,  perplexity=5)
embeddings_2d = tsne.fit_transform(embeddings)

plt.rcParams['font.sans-serif'] = ['Kaitt', 'SimHei']
plt.rcParams['axes.unicode_minus'] = False

color_list =  ['black'] * len(embeddings_2d[1:])
color_list.insert(0, 'red')

plt.scatter(embeddings_2d[:, 0], embeddings_2d[:, 1] , color=color_list )

for i in range(len(embeddings_2d)):
    plt.text(embeddings_2d[:,0][i], embeddings_2d[:,1][i]+2,  sentences[i]
,color=color_list[i] )

plt.show()      # 显示图表
```



a.基础信息表格

#不同的模型，有不同的维度，维度越多，能表达的信息量越多

#同一个模型，也有不同的尺寸，例如 m3e 中的 small, base, large

模型	向量维度	最大字符数	模型大小	支持特性	内存占用情况
multilingual-e5-small	384	512	0.47G	multi	0.9G
multilingual-e5-base	768	512	1.11G	multi	1.6G
multilingual-e5-large	1024	512	2.24G	multi	2.1G
jina-embeddings-v2-base-zh	768	8192	0.32G	multi	0.89G
bge-small-zh	512	512	95.8M	Ch	0.21G
bge-base-zh-v1.5	768	512	0.41G	Ch	0.89G
bge-large-zh-v1.5	1024	512	2.6G	Ch	1.25G
bge-m3	1024	8192	2.27G	multi	2.7G
m3e-base	768	512	820M	ch	0.89G
m3e-large	768	512	1.3G	Ch\En	1.62G
BCEmbedding	768	512	1.1G	ch、en	1.8G
acge_text_embedding	1024	1024	0.65G	ch	1.1G

b.模型分析

multilingual-e5

1.新的训练方法：E5采用了“EmbEddings from bidirEctional Encoder rEpresentations”这一创新方法进行训练，这意味着它不仅仅依赖传统的有标记数据，也不依赖低质量的合成文本对。

2.高质量的文本表示：E5能为文本提供高质量的向量表示，这使得它在多种任务上都能表现出色，尤其是在需要句子或段落级别表示的任务中。

3.多场景：无论是在Zero-shot场景还是微调应用中，E5都能提供强大的现成文本Embedding，这使得它在多种NLP任务中都有很好的应用前景

jina-embeddings-v2-base-zh

1.双语无缝对接

jina-embeddings-v2-base-zh 模型能够流畅处理中英文本，无论是作为搜索查询还是目标文档。中英文本中意义相近的内容都会被映射到相同的嵌入空间，为多语言应用奠定了坚实基础。

2.8k Token 超长文本支持

我们的模型支持长达 8K Token 的文本处理，这在开源向量模型中独树一帜，为处理更长的文本段落提供了显著优势。

3.高效紧凑的模型结构

jina-embeddings-v2-base-zh 模型以 322MB 的轻巧体积（包含 1.61 亿参数），输出维度为 768，能够在普通计算机硬件上高效运行，无需依赖 GPU，极大地提升了其实用性和便捷性。

bge-m3

1.特点：支持超过100种语言，具备领先的多语言、跨语言检索能力，全面且高质量地支撑“句子”、“段落”、“篇章”、“文档”等不同粒度的输入文本，最大输入长度为 8192，并且一站式集成了稠密检索、稀疏检索、多向量检索三种检索功能

2.不同检索方式

稠密检索：常用的向量检索方式，将文本映射为单个向量，通过向量相似度判断文本间的相关性。无需词汇匹配通用性强。

稀疏检索：例如经典的BM25检索算法，向量维度为整个词表，其中大部分为0，仅对文本中出现的单词计算出一个权重。有着更强的泛化能力和长文本建模能力。

多向量检索：对每个文本使用多个向量进行表示，代表性工作有Colbert。BGE-M3中采用了Colbert的交互机制计算相关性。多向量检索可以用于细粒度的检索和重排。

```
from FlagEmbedding import BGEM3FlagModel

model = BGEM3FlagModel('Shitao/bge-m3',  use_fp16=True)

sentences_1 = ["What is BGE M3?", "Defination of BM25"]
sentences_2 = ["BGE M3 is an embedding model supporting dense retrieval, lexical matching and multi-vector interaction.",
               "BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document"]

sentence_pairs = [[i,j] for i in sentences_1 for j in sentences_2]
print(model.compute_score(sentence_pairs))
# {
#     'colbert': [0.7796499729156494, 0.4621465802192688, 0.4523794651031494, 0.7898575067520142],
#     'sparse': [0.05865478515625, 0.0026397705078125, 0.0, 0.0540771484375],
#     'dense': [0.6259765625, 0.347412109375, 0.349853515625, 0.67822265625],
#     'sparse+dense': [0.5266395211219788, 0.2692706882953644, 0.2691181004047394, 0.563307523727417],
#     'colbert+sparse+dense': [0.6366440653800964, 0.3531297743320465, 0.3487969636917114, 0.6618075370788574]
# }
```

3.主要流程：多个索引分别检索top-k文本，把所有结果汇总再根据分数加权求和进行重排（或者使用bge-reranker模型进行重排）。pyserini进行的稀疏检索，faiss进行向量检索，colbert多向量用在重排阶段

Bcembedding

BCEmbedding由两部分构成：EmbeddingModel和RerankerModel。EmbeddingModel负责生成语义向量，主要用于语义搜索和问答任务；而RerankerModel则擅长优化语义搜索结果和精确排序。

1.双语和跨语种能力：基于有道翻译引擎的强大能力，`BCEmbedding` 实现强大的中英双语和跨语种语义表征能力。

2.RAG适配：面向RAG做针对性优化，可适配大多数相关任务，比如翻译，摘要，问答等。此外，针对 问题理解（query understanding） 也做了针对优化。

3.高效且精确的语义检索：`EmbeddingModel` 采用双编码器，可以在第一阶段实现高效的语义检索。`RerankerModel` 采用交叉编码器，可以在第二阶段实现更高精度的语义顺序精排。

4.用户友好：语义检索时不需要特殊指令前缀。也就是，你不需要为各种任务绞尽脑汁设计指令前缀。

5.有意义的重排序分数：`RerankerModel` 可以提供有意义的语义相关性分数（不仅仅是排序），可以用于过滤无意义文本片段，提高大模型生成效果。

acge_text_embedding

合合信息开发团队，国产自研文本向量化模型acge_text_embedding（以下简称“acge模型”）已经在业界权威的中文语义向量评测基准C-MTEB（Chinese Massive Text Embedding Benchmark）中获得了第一名。

同时在处理多样化任务时具有更高的灵活性，通用性更强，能够适用于多种文本处理任务。

3.完成本地知识库的需求分析

1.文档使用手册知识库构建

2.系统内文件内容 、dir、filename向量库构建

3.流程提问知识库构建