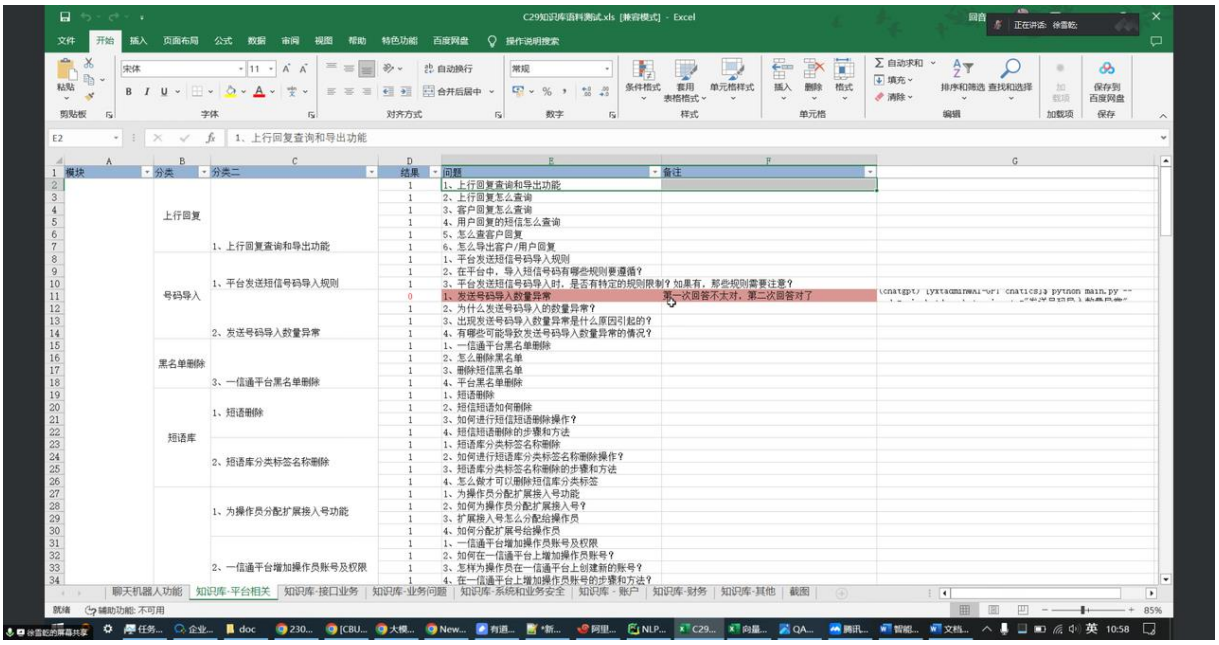
大模型 RAG

方向：

知识问答

1. 文档处理和数据处理：向量化---建立索引
2. 文档分割：切分成文档块，方便查找和索引
    a. 如果分块太小，某些问题就无法回答
    b. 分块太大，答案中就会产生噪音
    c. **如何分块？启发式（使用标点符号、段落结尾）和语义分块**
3. 问题与问题之间：先比较用户的问题和训练语料的问题



4. 问题与答案的匹配度：再从中获取最相关的 top n 条回答
5. 向量存储与向量检索：向量相似度搜索
6. 开源的和非开源的：Self-RAG, ElasticSearch, Redis，（大模型）百川，chatgpt

任务型对话

1. 意图识别：训练语料或提示语料，判断用户意图
2. 对话管理
3. 历史消息 是否要全部输入大模型，问题序列会前后影响
4. 工程实现：权限、数据查询、消息收发…

问答型对话

意图识别举例：判断用户意图，判断是否退出对话

历史消息：是否要全部输入大模型，问题序列会前后影响

韩老师总结：问题对（之前整理的）不整理 QA 从原文中整理出答案

相似度 问题和答案相似度

文本->向量 存储 查询

精彩手绘全解：RAG 技术，从入门到精通-腾讯云开发者社区-腾讯云(tencent.com)

1、 文本分块

对文档进行分块处理是在 RAG 管道中的外部知识来源的必要准备步骤，这可能会影响性能。它通常通过将长文档分解成更小的部分(但它也可以将更小的片段组合成连贯的段落)。

需要考虑的一个问题是分块技术的选择。例如，在 LangChain 中，不同的文本分割器通过不同的逻辑(如字符、标记等)来分割文档。这取决于拥有的数据类型。例如，如果输入数据是代码而不是 Markdown 文件，则需要使用不同的分块技术。

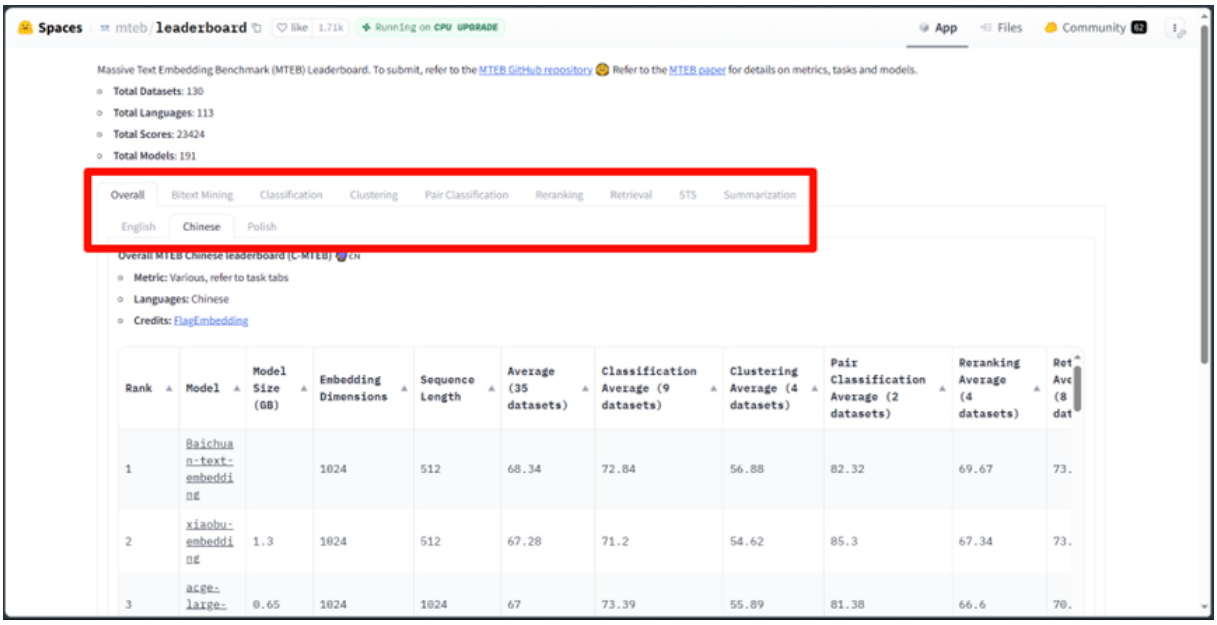
块的理想长度(chunk_size)取决于用例:如果用例是回答问题，可能需要更短的特定块，但是如果用例是总结，则可能需要更长的块。此外如果一个块太短，它可能没有包含足够的上下文，如果一个块太长，它可能包含太多不相关的信息。

初上面基本要求外，还需要考虑块之间的“滚动窗口”(重叠)，以引入一些额外的上下文。

2、 嵌入模型（文本向量化 Embedding）

嵌入模型是检索的核心。嵌入的质量严重影响检索结果。通常，生成的嵌入的维数越高，嵌入的精度就越高。

选择一个模型来嵌入切割后的块——有很多选择，例如像 bge-large 或 E5 嵌入系列这样的搜索优化模型——只需查看 **MTEB 排行榜上**[4]的最新更新。新建标签页 (huggingface.co)



在某些情况下，可能需要根据特定用例对嵌入模型进行微调以避免稍后出现域外问题。根据 LlamaIndex 进行的实验，微调嵌入模型可以使检索评估指标的性能提高 5-10%。

模型名（向量数据库 Embedding-关键特性-文档中心-腾讯云 (tencent.com)）	适用语言类型	Dimensions（维度）	最大 Token 数量	综合得分		
				Classification（分类）	Clustering（聚类）	Retrieval（
bge-base-zh（推荐）	中文	768	512	67.06	47.64	69.53
m3e-base	中文	768	512	67.52	47.68	56.91
text2vec-large-chinese	中文	1024	512	60.66	30.02	41.94
e5-large-v2	英文	1024	512	75.24	44.49	50.56

multilingual-e5-base	多语言	768	514	65.35	40.68	40.68

Rank	Model	Model Size (GB)	Embedding Dimensions	Sequence Length	Average (35 datasets)	Classification Average (9 datasets)	Clustering Average (4 datasets)	Pair Classification Average (2 datasets)	Reranking Average (4 datasets)	Retrieval Average (8 datasets)	STS Average (8 datasets)
1	Baichuan-text-embedding		1024	512	68.34	72.84	56.88	82.32	69.67	73.12	60.07
2	xiaobu-embedding	1.3	1024	512	67.28	71.2	54.62	85.3	67.34	73.41	58.52
3	acge-large-zh	0.65	1024	1024	67	73.39	55.89	81.38	66.6	70.96	58.02
4	gte-large-zh	0.65	1024	512	66.72	71.34	53.07	84.41	67.4	72.49	57.82
5	gte-base-zh	0.2	768	512	65.92	71.26	53.86	80.44	67	71.71	55.96
6	gte-base-align	0.44	768	512	65.82	71.26	53.86	80.44	67	71.53	55.72
7	tao-8k	0.67	1024	8192	65.5	69.05	49.04	82.68	66.38	71.85	58.66
8	IYun-large-zh	0.65	1024	1024	65.47	68.88	49.93	81.99	66.3	71.54	58.77
9	tao	0.65	1024	1024	65.14	69.05	49	82.68	66.39	70.26	58.66
10	stella-large-zh-v2	0.65	1024	1024	65.13	69.05	49.16	82.68	66.41	70.14	58.66
11	stella-large-zh	0.65	1024	1024	64.54	67.62	48.65	78.72	65.98	71.02	58.3
12	bge-large-zh-v1.5	1.3	1024	512	64.53	69.13	48.99	81.6	65.84	70.46	56.25
13	stella-base-zh-v2	0.21	768	1024	64.36	68.29	49.4	79.96	66.1	70.08	56.92
14	mist-zh	0.2	768	512	64.25	68.21	49.48	80.01	65.89	69.59	57.08
15	stella-base-zh	0.21	768	1024	64.16	67.77	48.7	76.09	66.95	71.07	56.54
16	piccolo-large-zh	0.65	1024	512	64.11	67.03	47.04	78.38	65.98	70.93	58.02
17	jina-embeddings-v2-				63.79	64.94	46.47	82.94	66.57	69.4	59.39

a) 非开源：[百川大模型-汇聚世界知识 创作妙笔生花-百川智能 (baichuan-ai.com)](baichuan-ai.com)

OpenAI 的 text-embedding-ada-002

b) 开源

i) BGE

[Embedding 开源模型重磅玩家 北京智源人工智能研究院最新 Embedding 模型发布 登顶 MTEB，免费商用授权！ | 数据学习者官方网站(Datalearner)]

BGE 模型的训练有 2 个阶段：分别是预训练阶段和微调阶段。

在预训练阶段，BGE 模型通过使用 RetroMAE 方法在中英文数据集上进行了预训练。其中英文数据集维 Pile、维基百科和 msmarco，而中文数据集则来自于 wudao。训练使用了 24 个 A100

（40G）。wudao 数据集：https://www.datalearner.com/blog/1051648084659749

在微调阶段，BGE 使用三元组数据（query, positive, negative）进行微调。使用了 48 个 A100（40G）的资源。目前，该数据没有开源，不过官方宣称未来将开源。

缺点：BGE 系列模型在 MTEB 榜单霸榜之后就引起了很多人的关注。对于我们来说，BGE 最大的好处是支持中文，且商用免费授权，是中文 embedding 模型中为数不多的优质选择。而它英文榜单霸榜也引起了很多人的使用，推特上也十分火爆。

不过，BGE 一个问题是它的输入序列只有 512，这在很多场景下可能不够用。而 OpenAI 的 text-embedding-ada-002 最高支持 8K 的序列长度。

基于 BGE 微调

lier007/xiaobu-embedding · Hugging Face

ii) acge-large-model（输入长度 1024）

aspire/acge-large-zh · Hugging Face

iii)gte-large-zh

The GTE models are trained by Alibaba DAMO Academy. They are mainly based on the BERT framework and currently offer different sizes of models for both Chinese and English Languages. The GTE models are trained on a large-scale corpus of relevance text pairs, covering a wide range of domains and scenarios. This enables the GTE models to be applied to various downstream tasks of text embeddings, including information retrieval, semantic textual similarity, text reranking, etc.

iv)IyunErin/IYun-large-zh · Hugging Face

无详细信息

v) stella-base-zh-v2 和 stella-large-zh-v2（输入长度 1024）

基于 piccolo 训练

完整训练思路及过程见：

stella 长文本编码模型 - 知乎 (zhihu.com)1

stella 长文本编码模型 - 知乎 (zhihu.com)2

1. 构造长于 512 的训练数据方法：

   1）开源数据

   数据集中长度大于 512 的数据，但是这些数据的关键信息都集中在了文本块的前半部分，模型只需要重点专注前半部分就可以判断是否匹配，根本不需要看后半部分。解决方案也非常简单：对于开源数据中的长数据，将其切分成句子，然后依次和 query 计算集合相似度，相似度最高的句子如果处于 passage 后半段那么就纳入训练数据，否则就以一定的比例舍弃。

   2）LLM 合成数据

   现在 LLM 的能力已经非常强大，所以还会考虑找一些长度大于 512 的无监督文本块，让 LLM 来生成相关 query。其中无监督文本块取自 wudao 语料库开源的 200GB，网

上可以自行下载。LLM 使用了 HFL 的 Chinese-Alpaca2-13B。具体构造方法为：抽取文本块后面的几个句子，然后模型生成对应的 query。prompt 比较简单，就不分享了。重点在于要让模型生成文本块后面的几个句子，这样在训练时才会逼着模型学习对文本块后面的文本进行编码

2. 如何确保训练模型时不会有灾难性遗忘

   我们希望在训练模型时，不要让模型遗忘掉原有的知识。stella 模型是在 piccolo 基础上训练的，piccolo 本身就拥有极强的文本编码能力。

   经过反复实验发现，向量模型的灾难性遗忘问题极其严重，学了新的，基本就把旧的忘了一大半。为了解决这个问题，主要采用了 2 个策略：

   1）Replay，即在训练数据中混合模型原有的训练数据和已有文本匹配数据

   2）EWC,Elastic Weights Consolidation，openai 的一篇论文，主要思想是对参数做一个约束，让新模型参数不至于偏离原模型太远。实际操作时，移除了需要在原始训练集计算的参数权重，转而使用固定值，权重设为了 10，对于 512-1024 的 position embedding，权重为 0，因为这块参数是需要进行更新学习的，因此不做约束。

   loss 非常简单，就是新旧模型参数的 mse 乘以权重

3. 评估长文本

   CMTEB 测试数据不能准确的评估长文本编码能力：

   1）长度大于 512 的过少

   2）即便大于 512，对于检索而言也只需要前 512 的文本内容

   因此本人搜集了相关开源数据并使用规则进行过滤，最终整理了 6 份长文本测试集,他们分别是：

   - CMRC2018，通用百科
   - CAIL，法律阅读理解
   - DRCD，繁体百科，已转简体
   - Military，军工问答
   - Squad，英文阅读理解，已转中文
   - Multifieldqa_zh，清华的大模型长文本理解能力评测数据

模型：

infgrad/stella-large-zh-v2 · Hugging Face

基于 stella 微调

amu/tao-8k · Hugging Face 拓展输入长度至 8k，但需将模型转换至 float32

vi)piccolo

sensenova/piccolo-large-zh · Hugging Face

piccolo 是一个通用 embedding 模型(中文)，由来自商汤科技的通用模型组完成训练。piccolo 借鉴了 E5 以及 GTE 的训练流程，采用了两阶段的训练方式。 在第一阶段中，我们搜集和爬取了 4 亿的中文文本对(可视为弱监督文本对数据)，并采用二元组的 softmax 对比学习损失来优化模型。 在第二阶段中，我们搜集整理了 2000 万人工标注的中文文本对(精标数据)，并采用带有难负样本的三元组的 softmax 对比学习损失来帮助模型更好地优化。 目前，我们提供了 piccolo-base-zh 和 piccolo-large-zh 两个模型。

vii) jina-embeddings-v2-base-zh

1. jinaai/jina-embeddings-v2-base-zh · Hugging Face

jina-embeddings-v2-base-zh 是支持中英双语的文本向量模型，它支持长达 8192 字符的文本编码。 该模型的研发基于 BERT 架构(JinaBERT)，JinaBERT 是在 BERT 架构基础上的改进，首次将 ALiBi 应用到编码器架构中以支持更长的序列。 不同于以往的单语言/多语言向量模型，我们设计双语模型来更好的支持单语言（中搜中）以及跨语言（中搜英）文档检索。

3、 索引算法（数据库结构）

	基于图的向量数据库	基于文档的向量数据库	键值型向量数据库
优点	1. 对复杂的数据关系和查询（data relationships and querying）非常有效。	1. 其灵活的模式允许存储不同数据结构的文件。	1. 读/写操作简单、高性能。
	2. 拥有表示和分析实体间关系的能力。	2. 本地支持面向文档的数据模型。	2. 具备可扩展性和高吞吐量，尤其适用于大型数据集。
	3. 非常适合数据元素相互关联的情况。	3. 易于与主流编程语言和开发框架集成。	3. 可根据唯一键高效缓存和检索数据。
缺点	1. 实现和管理简单的数据结构非常复杂。	1. 对复杂查询的支持有限，尤其是跨不同文件的查询。	1. 缺乏高级查询功能，数据之间的关系有限。
	2. 大型图（graphs）的存储和计算要求更高。	2. 不太适合高度标准化或关系型数据的应用场景。	2. 与关系数据库相比，数据操作能力有限。
	3. 对于不熟悉图形学理论的开发人员来说，学习曲线较长。	3. 涉及多个文件的复杂查询性能较慢。	3. 难以处理数据元素之间的复

a) 向量索引

一个为 10000+元素规模上的高效检索优化的索引是一个向量索引，如 faiss、nmslib 或 annoy，使用某种近似最近邻实现，如聚类、树或 HNSW 算法。

还有一些托管解决方案，如 OpenSearch 或 ElasticSearch，以及向量数据库，它们在后台处理第 1 步中描述的数据摄取管道，如 Pinecone、Weaviate 或 Chroma。根据选择的索引、数据和搜索需求，可以将元数据与向量一起存储，然后使用元数据过滤器来搜索某些日期或来源内的信息。

LlamaIndex 支持许多向量存储索引，但还支持其他更简单的索引实现，如列表索引、树索引和关键词表索引——我们将在融合检索部分讨论后者。

b) 分层索引

如果需要从许多文档中检索信息，您需要能够有效地在其中搜索，找到相关信息，并将其综合为带

有来源引用的单一答案。在大型数据库中做到这一点的有效方法是创建两个索引——一个由摘要组成，另一个由文档块组成，并分两步进行搜索，首先通过摘要筛选出相关文档，然后仅在这个相关组内搜索。

为了在规模上实现快速的相似性搜索，矢量数据库和矢量索引库使用近似最近邻(ANN)搜索而不是 k 最近邻(kNN)搜索。ANN 算法近似于最近邻，但可能不如 kNN 算法精确。

可以尝试不同的人工神经网络算法，比如 Facebook Faiss(聚类)、Spotify Annoy(树)、Google ScaNN(矢量压缩)和 HNSWLIB(接近图)。此外这些人工神经网络算法都有一些可以调优的参数，例如用于 HNSW 的 ef、efConstruction 和 maxConnections[1]。

如果数据量比较大，还可以为这些索引算法启用矢量压缩。与人工神经网络算法类似，矢量压缩会损失一些精度。但是根据矢量压缩算法的选择及其调优，也可以对其进行优化。

在实践中，这些参数已经由矢量数据库和矢量索引库的研究团队在基准测试实验期间进行了调整，而不是由 RAG 系统的开发人员进行调整。但是如果想尝试使用这些参数来挤出性能的最后一点，也是可以试试的。

c) 假设性问题和 HyDE

另一种方法是让 LLM 为每个块生成一个问题，并将这些问题嵌入向量中，在运行时针对这个问题向量索引进行查询搜索（在我们的索引中用问题向量替换块向量），然后在检索后路由到原始文本块，并将它们作为上下文发送给 LLM 以获得答案。这种方法通过查询与假设性问题之间更高的语义相似性，提高了搜索质量。

还有一种逆向逻辑方法称为 HyDE——让 LLM 给定查询生成一个假设性回应，然后使用其向量和查询向量来提高搜索质量。

d) 上下文检索

i) 句子窗口检索

在这个方案中，文档中的每个句子都分别嵌入，这提供了极高的查询与上下文余弦距离搜索的准确性。为了在找到最相关的单个句子后更好地推理所发现的上下文，我们通过在检索到的句子前后扩展 k 个句子的上下文窗口，然后将这个扩展的上下文发送给 LLM。

ii) 自动合并检索器（又称父文档检索器）

这里的想法与句子窗口检索非常相似——搜索更精细的信息片段，然后在将这些上下文提供给 LLM 进行推理之前扩展上下文窗口。文档被分割成较小的子块，这些子块引用较大的父块。

e) 融合检索或混合搜索

这是一个相对较老的想法，即从两个世界中各取所长——基于关键字的传统搜索（稀疏检索算法，如 tf-idf 或搜索行业标准 BM25）和现代语义或向量搜索，并将它们结合在一个检索结果中。这里唯

一的技巧是正确组合具有不同相似性得分的检索结果——这个问题通常通过使用倒数排名融合算法来解决，重新排列检索结果以获得最终输出。

在 **LangChain**[6]中，这是通过 Ensemble Retriever 类实现的，它结合了你定义的一系列检索器，例如 faiss 向量索引和基于 BM25 的检索器，并使用 RRF 进行重排。在 **LlamaIndex**[7]中这种做法也非常类似。

混合或融合搜索通常会提供更好的检索结果，因为它结合了两种互补的搜索算法，同时考虑了查询和存储文档之间的语义相似性和关键词匹配。

意图识别方法

意图识: Intention Recognition

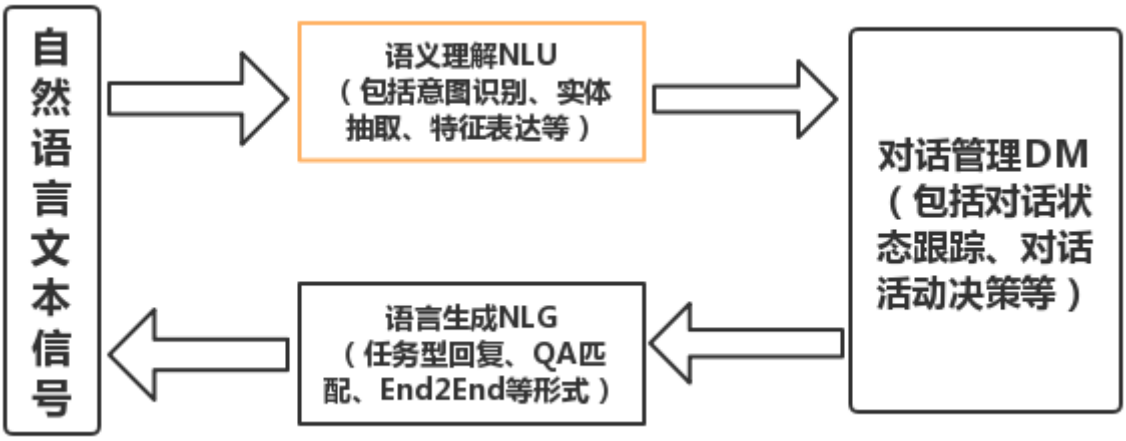
对于多轮任务型对话，首先需要理解用户主要说了啥，怎么说的以及对话的逻辑流程，并借助于对业务的理解以及对话文本的数据分析，抽象出对用户发言的语义理解定义，也即是语义理解模块。本篇主要是讲述意图识别的技术知识。意图识别通常需要使用自然语言处理技术，如分词、词性标注、句法分析、语义分析等。

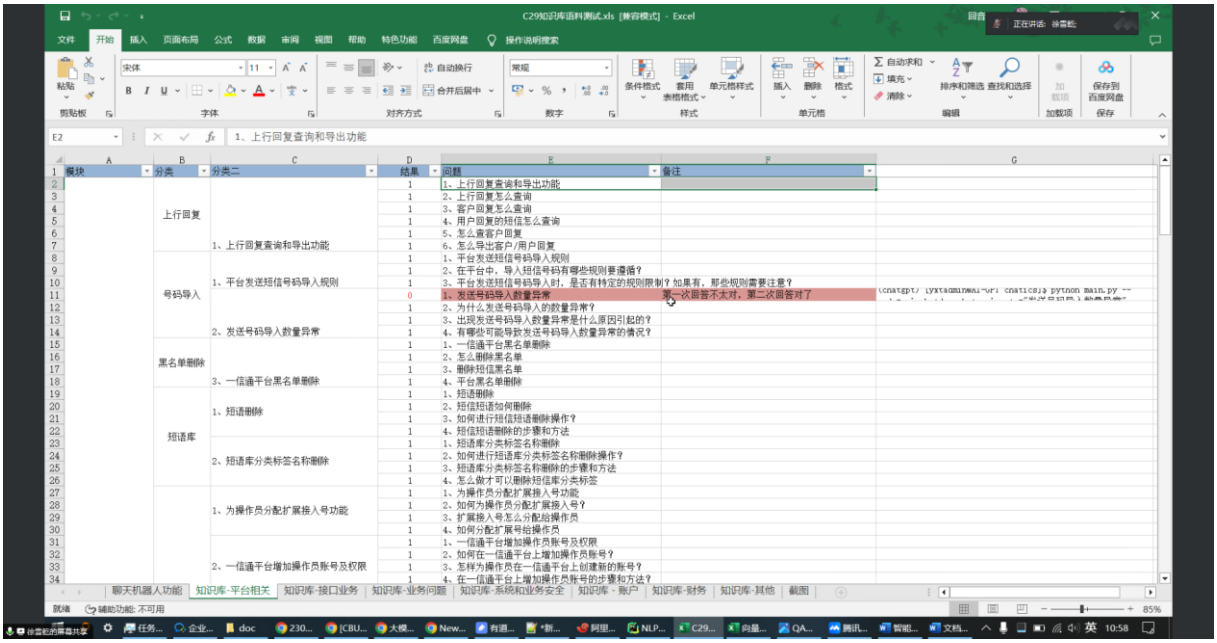
	样本	所属类别
0	我基金周一卖出什么时间能到账	faq
1	002887可以加仓吗？	advice
2	卖出收费吗	faq
3	能开港股吗	faq
4	000517连续三日最低点5.16。明天如果大盘下跌的话，应该会跌破吧	advice
5	集合竞价时如何买到热门股？	faq
6	可以分析下002497吗	advice
7	持有的000972后市如何操作	advice
8	昨天开的港股通今天能交易吗？	faq
9	南方航空还会再涨吗？	advice
10	什么是无量上涨？	edu
11	什么是市盈率？	edu
12	什么是布林线	edu
13	什么是移动平均线？	edu
14	什么是零股理论？	edu

在我们的业务场景中，意图细分成了 2 层的层级意图结构，也即话题意图以及在每个话题下的用户行为意图，也

可以理解为每种话题意图相当于一个 Chatbot。所以，话题意图的识别效果对后续流程影响较大。

话题意图识别属于文本分类任务，属于 NLU 任务之一。





2.2.2 意图识别结果

意图识别之后，我们能够得到3类问题，根据类别采用不同的决策，如果是任务型的，可以分配到具体的任务意图处理。

a) 问答型：例如"密码忘记怎么办？"→ 采用基于知识图谱构建 + 检索模型匹配方式

b) 任务型：例如"我想订一张明天从杭州到北京的机票"→ 意图决策 +slots filling 的匹配方式

c) 语聊型：例如"我心情不好"→ 检索模型与 Deep Learning 相结合的方式

话题意图分类模块构建

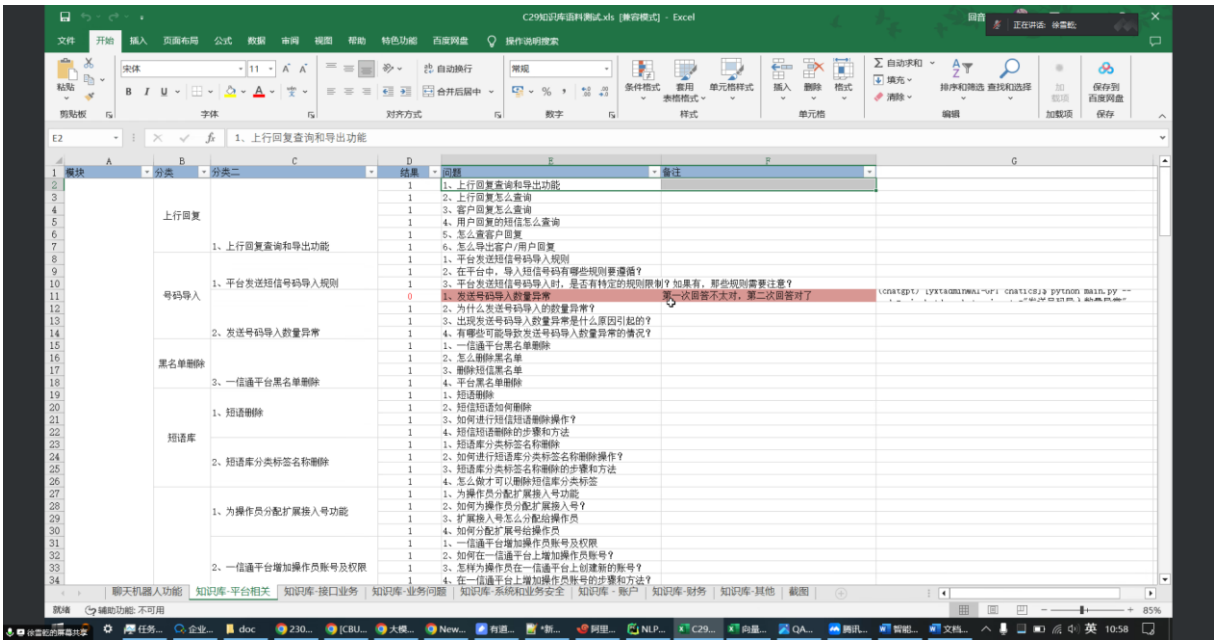
1. **分类模块的具体构建流程**

- 数据预处理：通过基于关键词和短语结构的正则表达式、自动化标签扩展模块这 2 种方法，清理不纯样本和修正错误样本标签最终得到 7 种 topic 类别（一般、安全模式、被盗、失

误、封号、信用、举报）的数据， 约 92w 左右。

- 通过新词发现模块把游戏名、装备名、地区名等名词加入到 Jieba 词表中；然后做带词性的切词，同时把句子切词后含有的游戏专有名词和游戏地区名替换为 GameName、AreaName 等通配符。这个 Trick 是为了减弱无关词对分类的影响，因为这些专有名词会在被盗、封号等类别中经常出现，影响分类效果。
- 基于全量数据使用 Word2vec 算法来预训练词向量、词性向量。
- 输入词编号和词性编号，加载预训练好的 Embedding 向量参数，训练基于两层 CNN+Highway 结构的文本分类模型，保存验证集合中准确率最好的模型。
- 模型多标签分类预测时，取预测概率值最高的类别，但当所有类别的预测概率都小于 0.3 时，判定为 common。经测试，此策略比单独取预测概率最高的类别返回，效果更好。因为话题模型是一个入口模型，决定着后续任务将交给哪一个类别的 Chatbot 处理，其中 common 类别是不触发具体的任务型 Chatbot；所以采用宁可放过也不愿意错判的策略。

（注：各个 topic 单独设计基于正则表达式的前置过滤规则，可配置过滤 badcase，便于安全运营，待积累一段时间的错判语料后，再更新线上的话题分类模型）

1. 

意图识别

1 词表列举法

通过词表直接匹配的方式来获取查询意图 例如查询词：德国[addr] 爱他美[brand] 奶粉[product] 三段[attr] 查询模式：[brand]+[product];[product]+[attr];[brand]+[product]+[attr]

Term Weight

引入词权重 Term Weight，给词不同的打分，根据分值就可以判断出核心词，进而可以应用到不同的场景。

比如，有一个商品的标题为“碗装保温饭盒套装”，通过 Term Weight 可以得到核心词为“饭盒”。

2 规则解析法

这种方法比较适用于查询非常符合规则的类别，通过规则解析的方式来获取查询的意图。比如： 北京到上海今天的机票价格，可以转换为[地点]到[地点] [日期] [汽车票/机票/火车票] [价格]。 1 吨等于多少公斤，可以转换为[数字] [计量单位]等于[数字] [计量单位]。

3 机器学习方法

意图识别其实可以看做是一个分类问题，采用机器学习或者文本分类方法，

可以统计出每种意图类别下面的常用词。【例如电商可以统计出类目词，产品词，品牌词，型号词，季节时间词，促销词等等】，例如： 地址(澳洲)，品牌词(爱他美)，产品词(奶粉)，属性词(三段)

对于用户输入的 query，根据统计分类模型计算出每一个意图的概率，最终给出查询的意图。

4 深度学习方法（CNN, LSTM, RCNN, C-LSTM, FastText）

文本特征+实时、离线用户本身的行为+用户本身相关的特征，通过深度学习方案构建模型，对用户意图进行预测。

整体的基本技术思路就是将行为因子与文本特征分别进行 Embedding 处理，通过向量叠加之后再进行多分类或者二分类处理。

c-LSTM

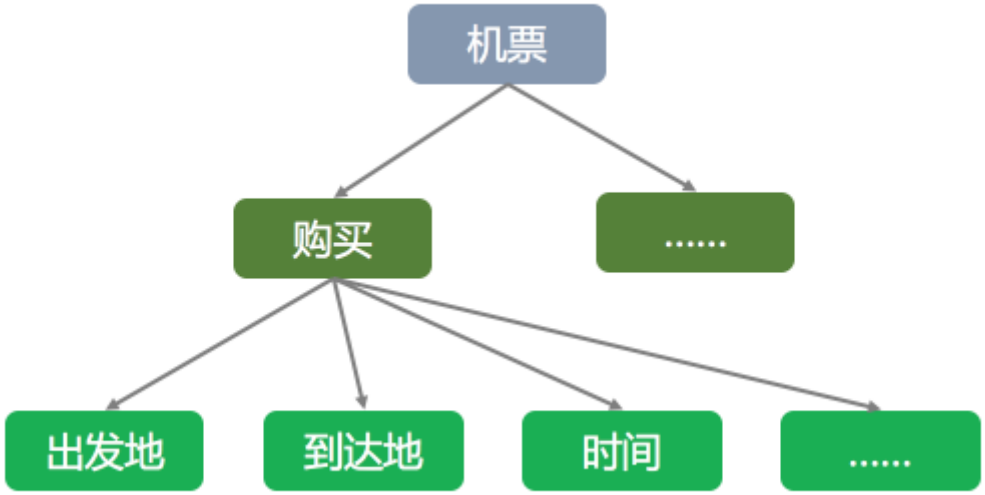
FastText

2. 意图识别结果

意图识别之后，我们能够得到 3 类问题，根据类别采用不同的决策，如果是任务型的，可以分配到具体的任务意图处理。

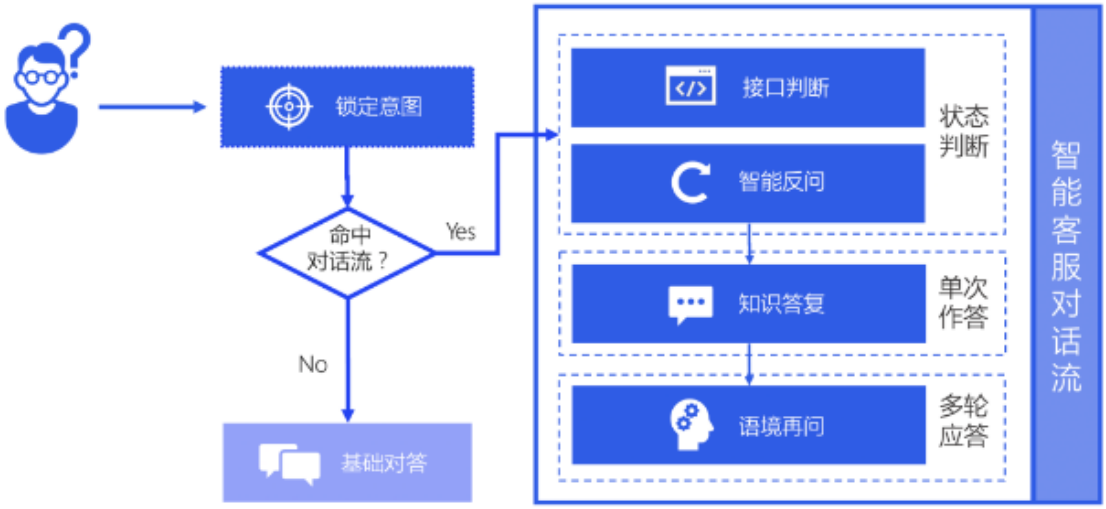
a) 问答型：例如“密码忘记怎么办？”→ 采用基于知识图谱构建 + 检索模型匹配方式

b) 任务型：例如“我想订一张明天从杭州到北京的机票”→ 意图决策 +槽位填充 的匹配方式。槽位填充通常需要使用对话管理技术，如对话状态跟踪、对话策略生成等。



c) 语聊型：例如“我心情不好”→ 检索模型与 Deep Learning 相结合的方式，考察的是大模型的能力。

3. 多轮对话策略



-

在线强化学习

- 确定意图边界和意图切换逻辑：在多轮对话中，需要维护一个意图堆栈，**确定当前的意图和意图之间的转移逻辑。**
- 维护当前意图集合：在对话过程中，用户可能会不断添加或修改意图的子意图，因此需要维护一个当前识别的意图集合

4. 开源的意图识别模型

基于 BERT：https://github.com/monologg/JointBERT

- 同时预测意图和槽位填充

数据集：ATIS/Snips

内外小蜜

知识增强：一般的知识增强会用开源知识图谱，但是企业内部的专有名词没办法在开源的知识图谱里面找到对应的实体，所以我们就从内部找知识。刚好阿里内外有一个知识卡片搜索的功能，每个知识卡片对应的是一个内网的产品，它和内外小蜜的领域是高度相关的，像这里面的欢行、爱豆都能找到相关的知识卡片，所以就把企业知识卡片作为知识来源。

采用 BERT 似乎不可取

短文本分类问题：**规则匹配—>机器学习—>传统深度学习—>前沿深度学习**

fasttext

TextCNN: TextCNN 相比于 FastText，利用 CNN (Convolutional Neural Network）来提取句子中类似 n-gram 的关键信息，且结构简单，效果好

TextRNN: 尽管 TextCNN 能够在很多任务里面能有不错的表现，但 CNN 最大的问题

是固定 filter_size 的视野，一方面无法建模更长的序列信息，另一方面 filter_size 的超参调节很繁琐。CNN 本质是做文本的特征表达工作，而自然语言处理中更常用的是递归神经网络（RNN, Recurrent Neural Network），能够更好的表达上下文信息。具体在文本分类任务中，Bi-directional RNN（实际使用的是双向 LSTM）从某种意义上可以理

解为可以捕获变长且双向的的 “n-gram” 信息。

C-LSTM

Transformer+KD

有些短文本分类任务可以转化为**语义匹配**任务，即短文本和标签文本做语义匹配

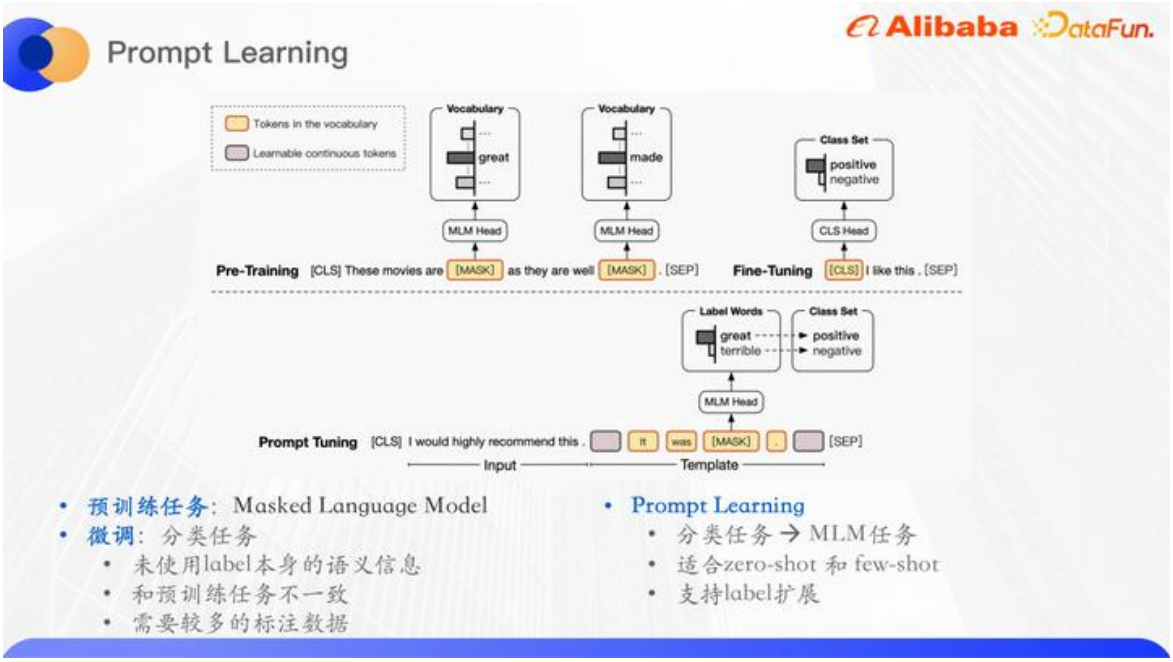
- NB：使用了样本属性独立性的假设，所以如果样本属性有较强的关联时其效果不好；
- KNN：非参数化，无须训练，但计算和存储成本与训练集大小成正比
- SVM：效果很好，在集成学习和 NN 之前几乎在分类领域处于统治地位，但在样本数较大时的存储和计算成本过高
- 决策树：可解释性好，基于决策路径直接来写规则
- 集成方法：xgboost/lightgbm 等

5. 少样本情境下的意图识别

结合大模型，为了提高智能客服的意图识别准确度，考虑一下优化方法：

- 数据增强：对已有的用户 query 进行词替换、词序替换、插入新词，生成新的训练样本
- 迁移学习：可以借助大模型的泛化能力。BERT-->NLU
- 元学习：学习如何学习，通过学习大量不同任务的经验，使得模型能够从少量样本中快速学习和泛化。Task invariant knowledge(entity recognition, relation extraction, slot filling, sentiment analysis, text classification, semantic role labeling)

以采购商城为例，采购商城有自己的商品类目体系，每个商品在上架之前会挂载到一个商品类目下。为了提高商城搜索的准确率，需要把 query 也预测到具体的类目，再根据这个类目调整搜索排序结果，也可以根据类目结果，在界面上展示子类目导航和相关搜索。目预测需要人工标注的数据集，但是采购领域，标注的成本比较高，所以从小样本分类的角度来解决这个问题。



在 zero shot 场景下分类的效果，预训练模型用的是 Bert base，一共是 30 个类，zero shot 已经能达到 16% 的准确率了。在 ten shot 数据集上进行训练，几种模板最高能达到 56% 的准确率

参考：

[基于知识增强和预训练大模型的 Query 意图识别 - 知乎 (zhihu.com)](https://www.zhihu.com/tardis/zm/art/443782891?source_id=1005)

[大模型+知识库/数据库问答实践过程的经验汇总（三） - 知乎 (zhihu.com)](https://www.zhihu.com/tardis/zm/art/443782891?source_id=1005)

https://www.zhihu.com/tardis/zm/art/443782891?source_id=1005

https://zhuanlan.zhihu.com/p/596112080

挑战

向量数据库（搭建方法）

矢量数据库对比和选择指南-腾讯云开发者社区-腾讯云(tencent.com)

矢量数据库的主要优点是它能够根据数据的矢量接近度或相似性快速准确地定位和检索数据。这允许基于语义或上下文相关性的搜索，而不是像传统数据库那样仅仅依赖于精确匹配或设置标准。

开源向量数据库&工具库

向量数据库｜一文全面了解向量数据库的基本概念、原理、算法、选型-腾讯云开发者社区-腾讯云(tencent.com)

1、纯矢量数据库

纯矢量数据库是专门为存储和检索矢量而设计的。包括 Chroma, LanceDB, Marqo, Milvus/ Zilliz, Pinecone, Qdrant, Vald, Vespa, Weaviate 等。数据是基于对象或数据点的向量表示来组织和索引。这些向量可以是各种类型数据的数字表示，包括图像、文本文档、音频文件或任何其他形式的结构化或非结构化数据。

优点

- 利用索引技术进行高效的相似度搜索
- 大型数据集和高查询工作负载的可伸缩性
- 支持高维数据
- 支持基于 HTTP 和 json 的 api
- 原生支持向量运算，包括加法，减法，点积，余弦相似度

缺点

纯矢量数据库可以存储矢量和一些元数据，不支持很多非结构化数据。

有限或没有 SQL 支持:纯矢量数据库通常使用自己的查询语言，这使得很难对矢量和相关信息运行传统的分析，也很难将矢量和其他数据类型结合起来。

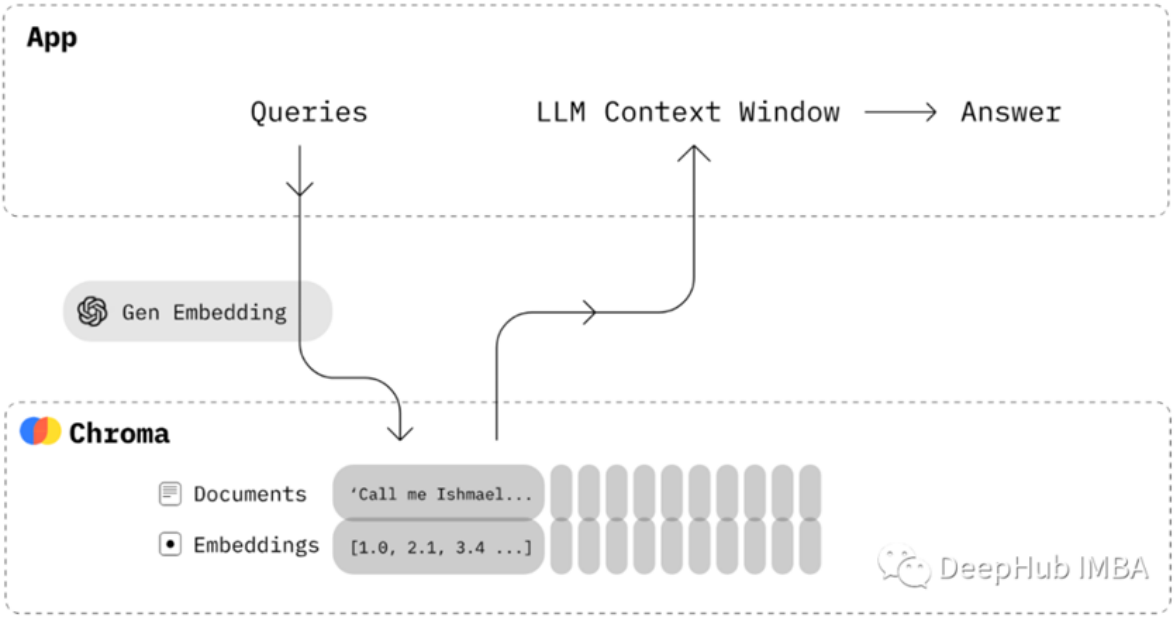
没有完整的 CRUD:纯矢量数据库并不是真正为创建、更新和删除操作而设计的。所以必须首先对数据进行矢量化和索引，这些数据库的重点是获取向量数据，并基于向量相似度查询最近邻，而索引是很耗时的。索引矢量数据计算量大、成本高、耗时长。这使得基本上无法进行实时的操作。例如，Pinecone 的 IMI 索引(反向多索引，人工神经网络的一种变体)会产生存储开销，并且是计算密

集型。它主要是为静态或半静态数据集设计的，如果经常添加、修改或删除向量，基本上不太可能。而 Milvus 使用的索引被称为产品量化和分层可导航小世界(HNSW)，这是一种近似的技术，在搜索准确性和效率之间进行权衡。它的索引需要配置各种参数，使用不正确的参数选择可能会影响搜索结果的质量或导致效率低下。

功能性不强:许多矢量数据库在基本特性上严重落后，包括 ACID 事务、灾难恢复、RBAC、元数据过滤、数据库可管理性、可观察性等。这可能会导致严重的业务问题，要解决这些问题，则需要我们自己来处理了这会导致开发量大增。

Chroma

https://zhuanlan.zhihu.com/p/665715823



Chroma 是开源嵌入数据库。通过为 LLM 提供可插入的知识，事实和技能，使构建 LLM 应用程序变得容易，可以轻松地管理文本文档，将文本转换为嵌入，并进行相似度搜索。

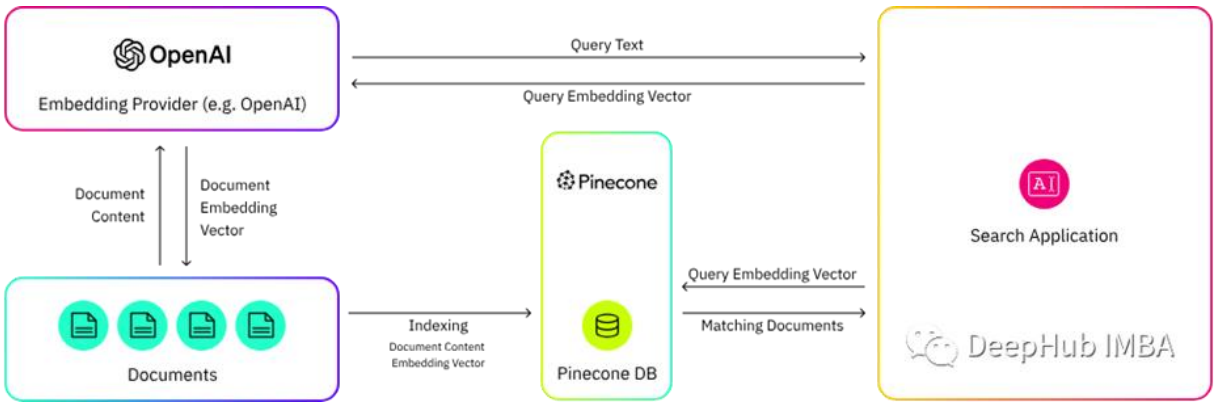
主要特点:

功能丰富:查询、过滤、密度估计和许多其他功能

LangChain (Python 和 javascript)， LlamaIndex 都支持

在 Python notebook 中运行的相同 API 可扩展到生产集群

Pinecone



Pinecone 是一个可以托管向量数据库平台。也就是说有背后的而商业公司，有免费使用方案。

Pinecone 的主要特点包括:

支持全托管服务

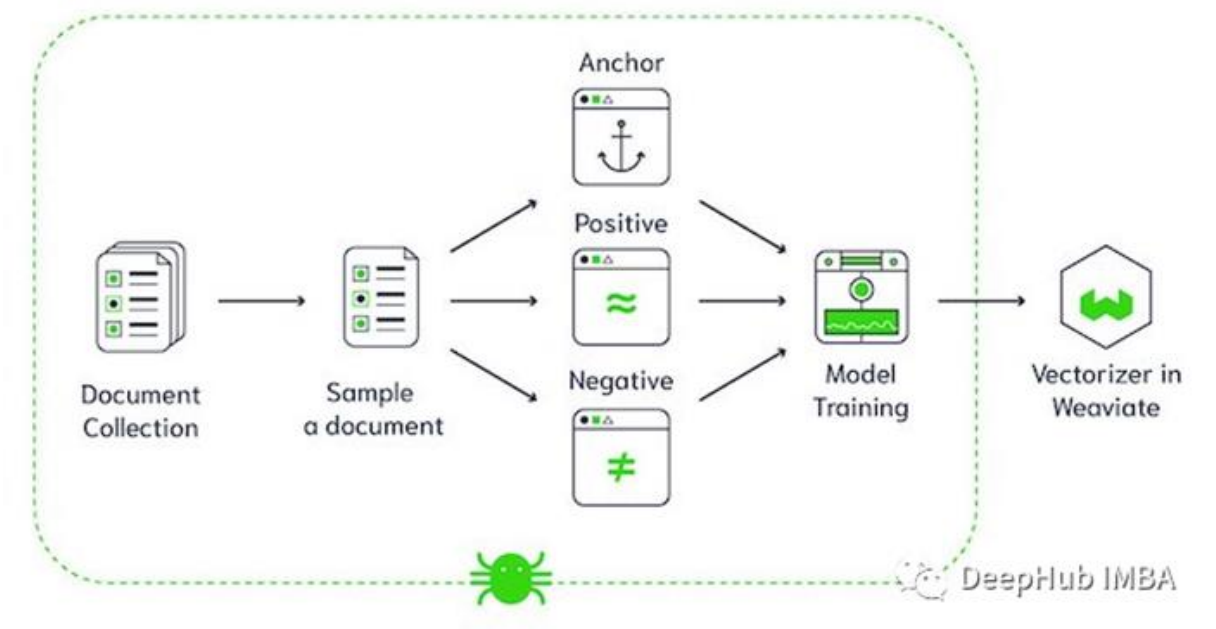
高度可伸缩

实时数据摄取

低延迟的搜索

与 LangChain 集成

Weaviate



Weaviate 是一个开源向量数据库。它可以无缝扩展到数十亿个数据对象。Weaviate 的一些关键特性是:

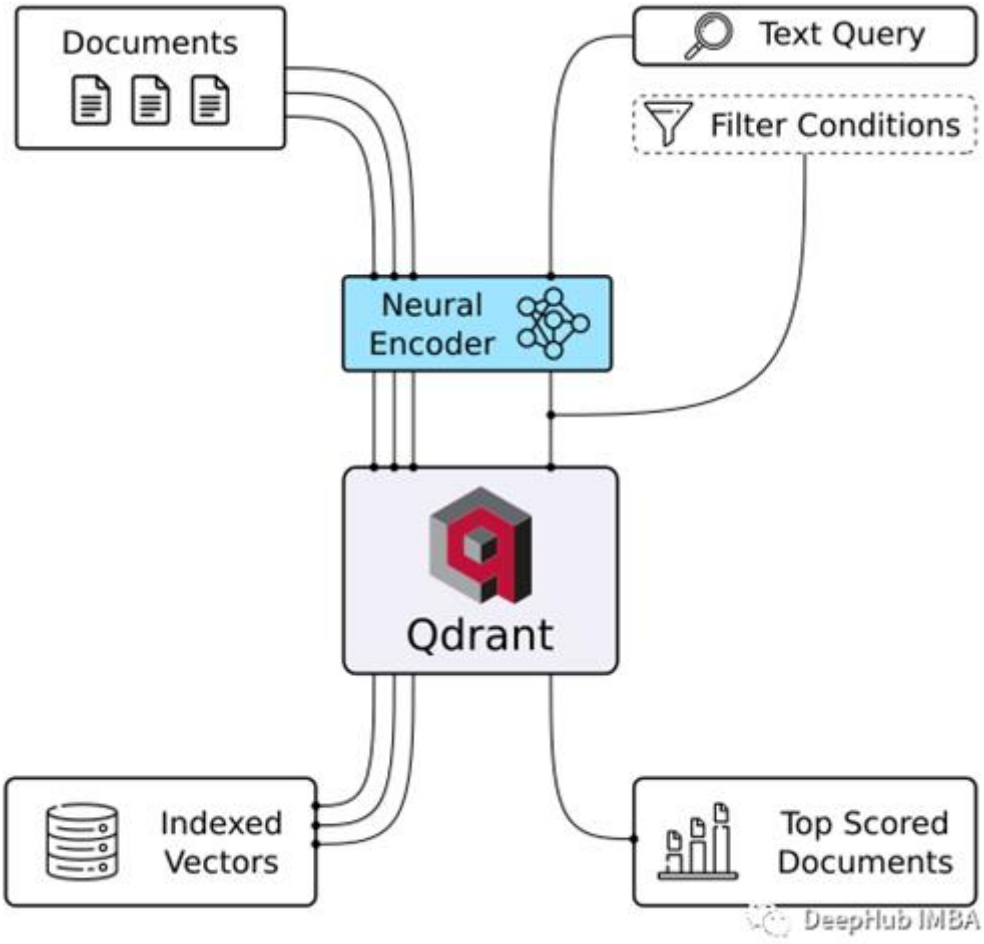
速度：Weaviate 可以在几毫秒内从数百万个对象中快速搜索出最近的 10 个邻居。

灵活性：使用 Weaviate，可以在导入或上传自己的数据时对数据进行矢量化，可以利用与 OpenAI, Cohere, HuggingFace 等平台集成的模块。

快速部署：从原型到大规模生产，Weaviate 都强调可伸缩性、复制和安全性。

搜索扩展:除了快速矢量搜索，Weaviate 还提供推荐、摘要和神经搜索框架集成。

Qdrant



Qdrant 可以作为 API 服务运行，支持搜索最接近的高维向量。使用 Qdrant，可以将嵌入或神经网络编码器转换为应用程序，用于匹配，搜索，推荐等任务。以下是 Qdrant 的一些关键功能:

通用的 API：提供 OpenAPI v3 规范和各种语言的现成客户端。

速度和精度：使用自定义 HNSW 算法进行快速准确的搜索。

先进的过滤方法：允许基于相关矢量有效载荷的结果过滤。

不同的数据类型：支持字符串匹配、数字范围、地理位置等。

可伸缩性：具有水平扩展功能的云原生设计。

效率：内置 Rust，通过动态查询规划优化资源使用。

2、 算法加持的开源矢量库

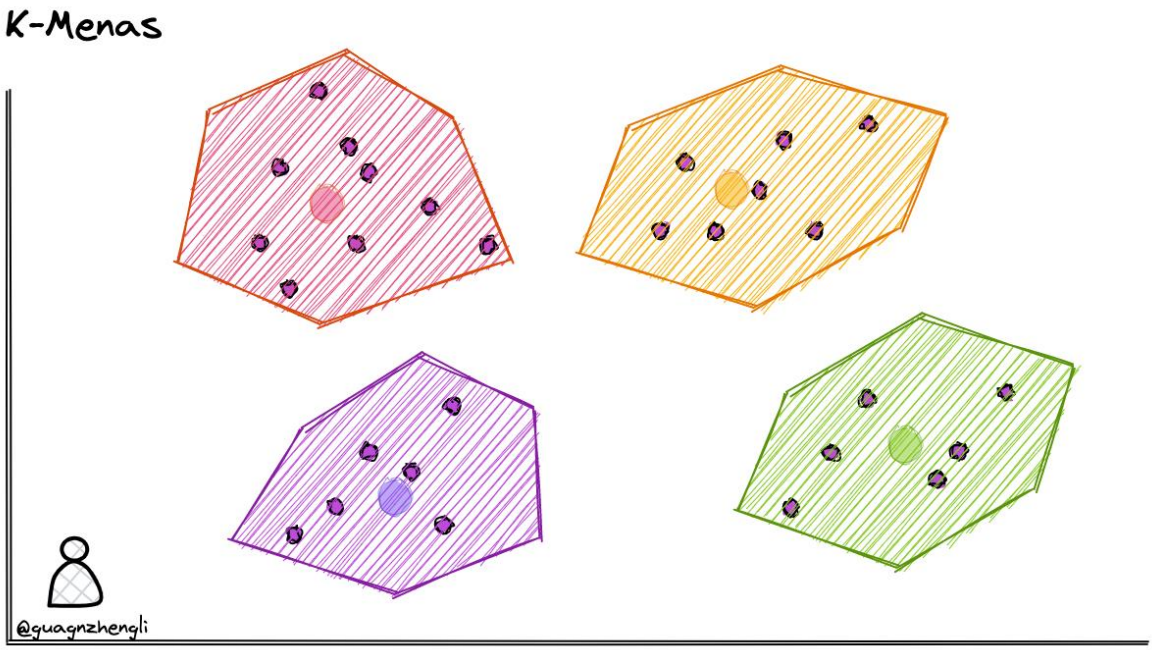
高效的搜索算法有很多，其主要思想是通过两种方式提高搜索效率：

1. 减少向量大小——通过降维或减少表示向量值的长度。

2. 缩小搜索范围——可以通过聚类或将向量组织成基于树形、图形结构来实现，并限制搜索范围

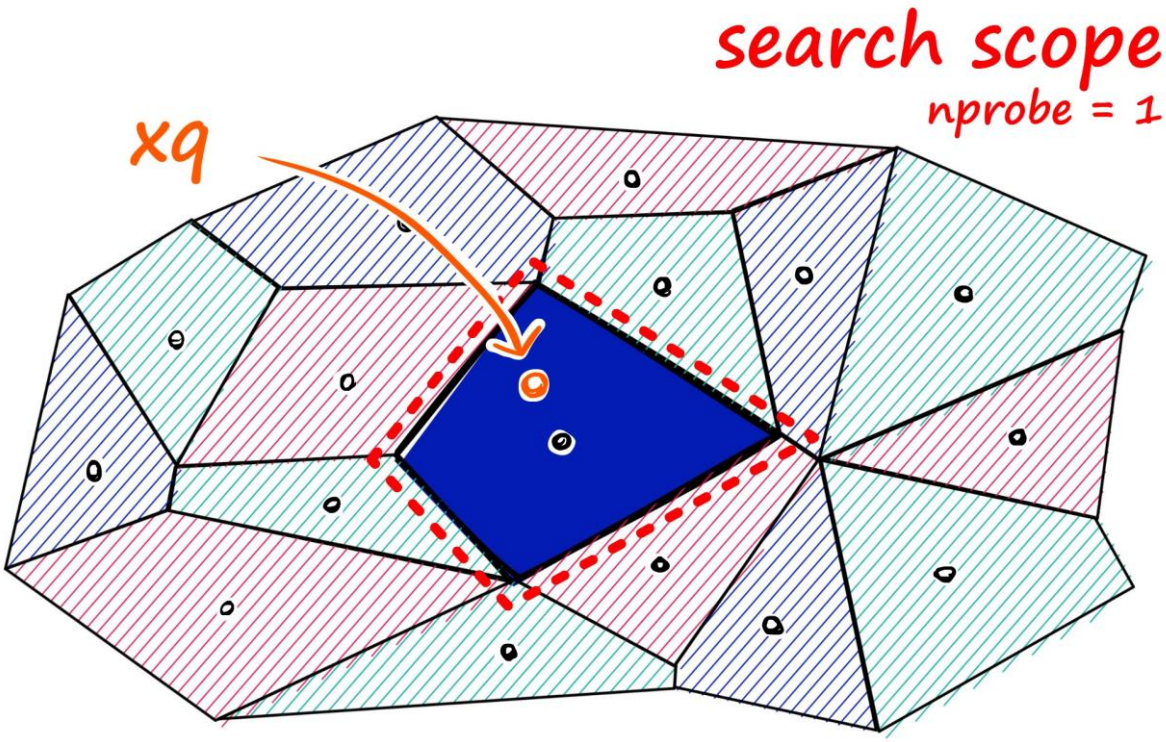
仅在最接近的簇中进行，或者通过最相似的分支进行过滤。

运用聚类——近似最相邻：K-Means、 Faiss、HNSW



对于许多开发者来说，Faiss、Annoy 和 Hnswlib 等开源矢量库是一个很好的起点。

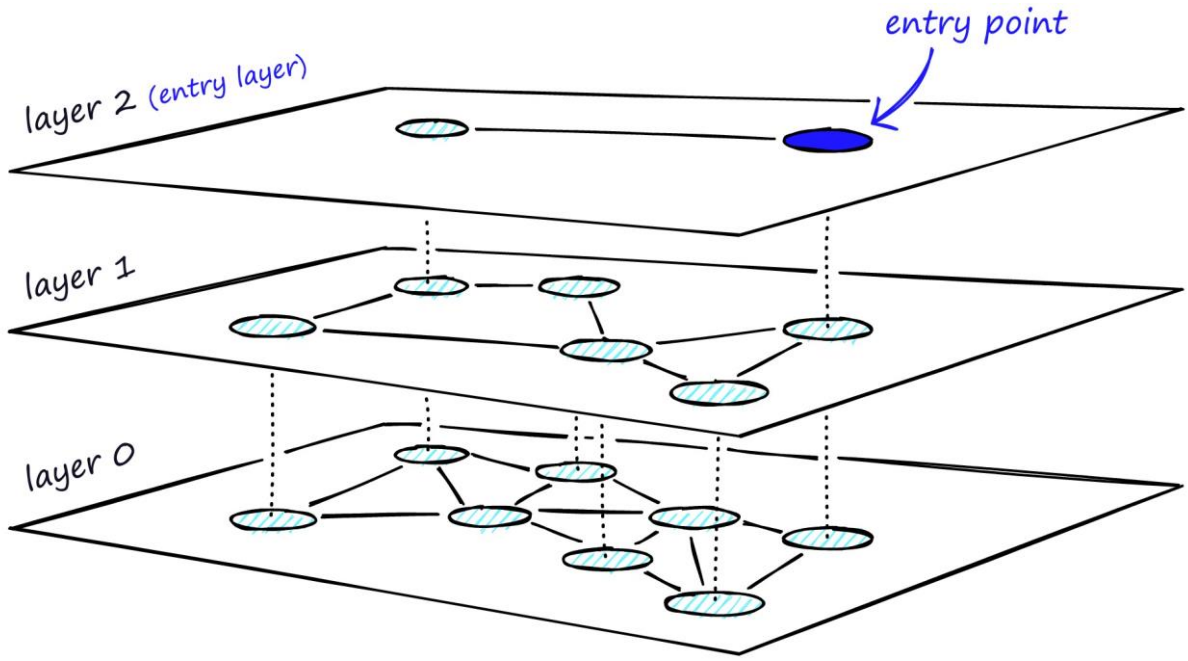
Faiss 是一个用于密集向量相似性搜索和聚类的库。



Annoy (Approximate Nearest Neighbors Oh Yeah)是一个用于人工神经网络搜索的轻量级库。

Hnswlib 是一个实现 HNSW ANN 搜索算法的库。这种方法的基本思想是每次将向量加到数据

库中的时候，就先找到与它最相邻的向量，然后将它们连接起来，这样就构成了一个图。当需要搜索的时候，就可以从图中的某个节点开始，不断的进行最相邻搜索和最短路径计算，直到找到最相似的向量。



Product Quantization (PQ)

在高维坐标系中，还会遇到维度灾难问题，具体来说，随着维度的增加，数据点之间的距离会呈指数级增长，这也就意味着，在高维坐标系中，需要更多的聚类中心点将数据点分成更小的簇，才能提高分类的质量。否者，向量和自己的聚类中心距离很远，会极大的降低搜索的速度和质量。

解决方法：解决这个问题的方法是将向量分解为多个子向量，然后对每个子向量独立进行量化，比如将 128 维的向量分为 8 个 16 维的向量，然后在 8 个 16 维的子向量上分别进行聚类，因为 16 维的子向量大概只需要 256 个聚类中心就能得到还不错的量化结果，所以就可以将码本的大小从 2^64 降低到 8 * 256 = 2048 个聚类中心，从而降低内存开销。

Locality Sensitive Hashing (LSH)

局部敏感哈希（Locality Sensitive Hashing）也是一种使用近似最近邻搜索的索引技术。它的特点是快速，同时仍然提供一个近似、非穷举的结果。LSH 使用一组哈希函数将相似向量映射到“桶”中，从而使相似向量具有相同的哈希值。这样，就可以通过比较哈希值来判断向量之间的相似度。

优点

- 快速近邻搜索
- 为高维构建
- 支持面向人工神经网络的索引结构，包括倒排文件，产品量化和随机投影

•支持推荐系统、图像搜索和自然语言处理的用例

•SIMD(单指令，多数据)和 GPU 支持，加快向量相似度搜索操作

缺点

•维护和集成麻烦

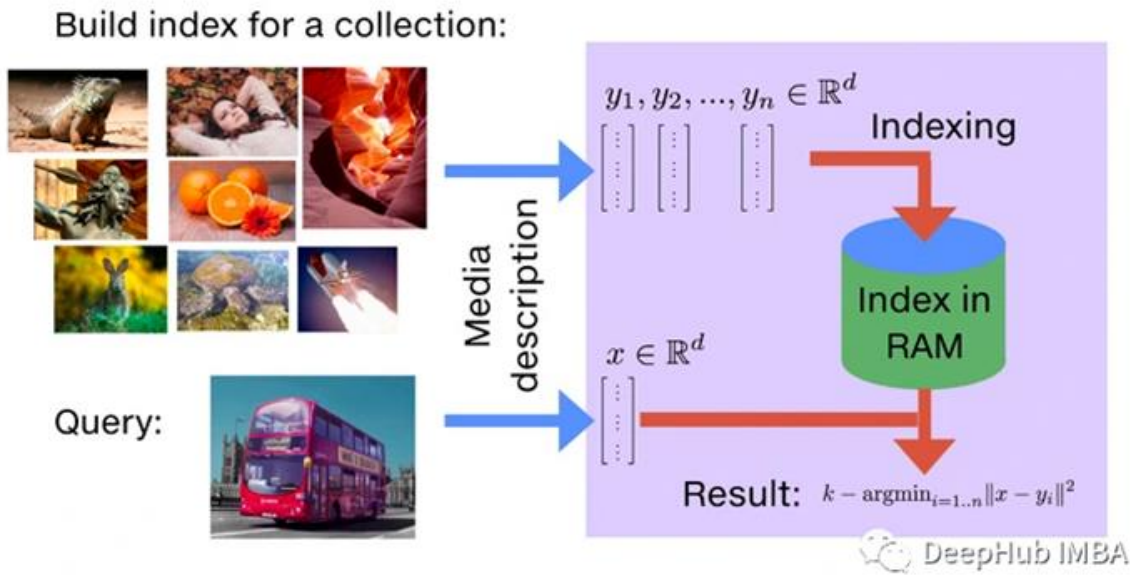
•与精确方法相比，可能会牺牲搜索准确性

•需要自己部署和维护：需要你构建和维护复杂的基础设施，为应用程序需求提供足够的 CPU、GPU 和内存资源。

•对元数据过滤、SQL、CRUD 操作、事务、高可用性、灾难恢复以及备份和还原的支持有限或不支持

他们之所以称为库（或者包）而不是数据库是因为它们只提供了很少的但是却非常专业功能，如果你想入门学习或者做一个简单的 demo，它们都是很好开始，但不建议直接应用到生产中。

Faiss（向量处理工具，用于搭建）



Faiss 是一个用于快速搜索相似性和密集向量的聚类的开源库。它包含能够在不同大小的向量集中搜索的算法，甚至可以处理那些超过内存容量的向量集。此 Faiss 还提供了用于评估和调整参数的辅助代码。

虽然它主要是用 c++编写的，但它完全支持 Python/NumPy 集成。它的一些关键算法也可用于 GPU 执行。Faiss 的主要开发工作由 Meta 的基础人工智能研究小组承担。

应用示例：

FAISS + SBERT 实现的十亿级语义相似性搜索-知乎 (zhihu.com)

中文向量数据库

腾讯云

向量数据库_大模型知识库_向量数据存储_向量数据检索-_腾讯云_(tencent.com)

总结

向量数据库	URL	GitHub Star
chroma	https://github.com/chroma-core/chroma	7.4K
milvus	https://github.com/milvus-io/milvus	21.5K
pinecone	https://www.pinecone.io/	✕
qdrant	https://github.com/qdrant/qdrant	11.8K
typesense	https://github.com/typesense/typesense	12.9K
weaviate	https://github.com/weaviate/weaviate	6.9K

3、文本匹配、相似度计算

1、文本匹配/文本相似度使用场景

首先我们将对话系统从分成两层：

1、 意图识别层：识别语言的真实意图，将意图进行分类并进行意图属性抽取。意图决定了后续的领域识别流程，因此意图层是一个结合上下文数据模型与领域数据模型不断对意图进行明确和推理的过程；

2、 问答匹配层：对问题进行匹配识别及生成答案的过程。在阿里小蜜的对话体系中我们按照业务场景进行了 3 种典型问题类型的划分，并且依据 3 种类型会采用不同的匹配流程和方法：

意图识别之后，我们能够得到 3 类问题，根据类别采用不同的决策，如果是任务型的，可以分配到具体的任务意图处理。

a) 问答型：例如“密码忘记怎么办？”→ 采用基于知识图谱构建 + 检索模型匹配方式

b) 任务型：例如“我想订一张明天从杭州到北京的机票”→ 意图决策 +slots filling 的匹配方式

检索模型基本处理过程

1、预处理 query ->分词-> 纠错->标准化->文本特征提取->query 改写[同义词]

2、算法召回

3、计算语句和候选句的相似度 s(q,q')，并排序

4、排序调整

深度学习 deepQA 的基本处理过程

1、query -> 分词->纠错->标准化->文本特征提取->query 改写[同义词]

2、命名实体识别

3、再进行用户意图识别

4、根据用户意图，当前轨迹，当前问题，检索答案候选集，生成问题-答案对

5、特征抽取，计算问题答案相似度

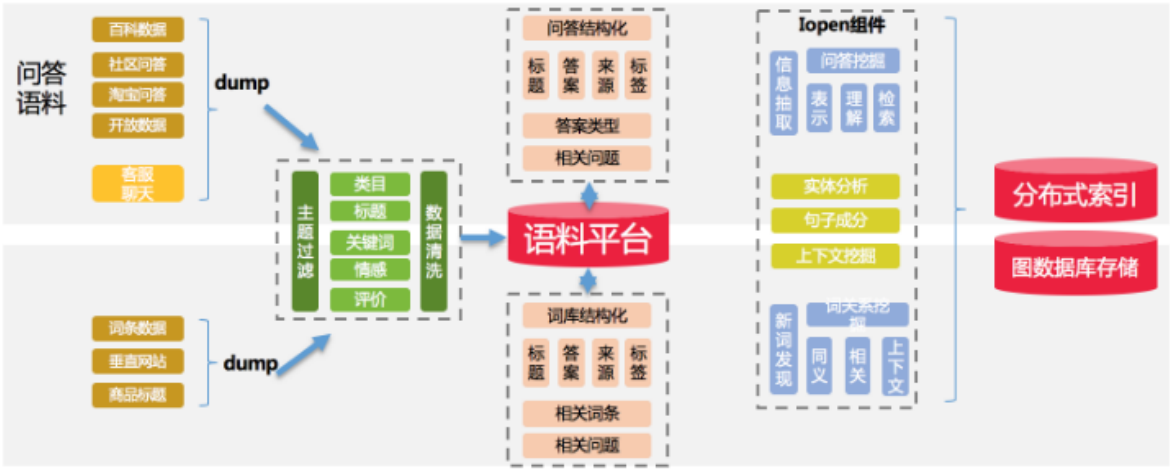
6、提取相似度最高的问题答案

DeepQA 实质上就是从海量的候选集合里面筛选符合用户问题的答案

举例：阿里小蜜的解决方案=知识图谱构建 +检索模型匹配方式

特点：有领域知识的概念，且知识之间的关联性高，并且对精准度要求比较高.

知识图谱的构建: 一个是实体维度的挖掘，一个是短句维度进行挖掘，通过在淘宝平台上积累的大量属于以及互联网数据，通过主题模型的方式进行挖掘、标注与清洗，再通过预设定好的关系进行实体之间关系的定义最终形成知识图谱。



2、文本匹配、相似度计算方法

对文本匹配方法来说，我们一般可以使用两种：无监督和有监督。

2.1、无监督文本匹配

首先我们来谈一下无监督的方法。

文本表征和相似函数的度量。

文本表征指的是我们将文本表示为计算机可以处理的形式，更准确了来说是数字化文本。而这个数字化文本，必须能够表征文本信息，这样才说的通。

相似函数的度量就是你选择何种函数对文本相似度进行一个判定，比如欧氏距离，余弦距离，Jacard 相似度，海明距离等等

文本表示模型		相似度度量方法
文本切分粒度	特征构建方法	
原始字符串 ngram 词语 句法分析结果 主题模型	TF TF-IDF 句向量 词向量 Simhash	最小编辑距离 欧氏距离 余弦距离 杰卡德相似度 海明距离
		分类器

有监督方法，就是用朴素贝叶斯分类器之类的有监督模型来判断文本相似性或者计算相似度。这类方法要求有一定数量的标注语料，构建的代价比较高；由于训练语料通常无法做得很大，模型的泛化性不够，实际用起来会有点麻烦；距离计算环节的复杂度会比较高。

无监督方法，就是用欧氏距离等方法，直接计算文本之间的距离或者相似度。这类方法的特点是：不需要标注语料，特征工程或者参数估计可以使用很大的数据；很多方法对语言的依赖比较小，可以应对多语种混杂的场景；距离计算环节复杂度较低。

1、TF-IDF

TF-IDF(Term Frequency-Inverse Document Frequency, 词频-逆文件频率)是相似度检索的常用加权技术。TF-IDF 是一种统计方法，用以评估一字词对于语料库中的其中一份文本的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

词频计算方式

$$IDF_i = \log \frac{|D|}{1 + |j : t_i \in d_j|}$$

逆文档计算方式

$$TF\text{-}IDF = TF \cdot IDF$$

TF-IDF 计算公式

因为 TF-IDF 的计算方法不涉及到语义只是利用了统计信息，所以一般用于召回。

优点：计算速度快，可用于大规模数据召回 简单易懂

缺点：词频未考虑上限，对于文档长度没有考虑，只有统计信息，没有语义信息

2、BM25

BM25 是上面 TF-IDF 的升级版算法，主要是针对于 TF-IDF 的缺点进行了一定的修复，著名的开源高性能搜索框架 Elastic Search 的默认算法，Elastic Search 在搜索和检索领域应用极广。

优点：计算速度快，可用于大规模数据召回 简单易懂

缺点：只有统计信息，没有语义信息

3、词向量相加平均句向量

现在词向量的种类有很多种，例如 word2vec、fasttext、glove、腾讯词向量等，词向量对比上面的 BM25 和 TF-IDF 的优点就是有学到语义信息。通过词向量相加平均来形成句向量计算相似度。

优点：

词向量在大量语料中学到了语义

训练成本低，训练简单

缺点：

未考虑到单词重要性和语序

句子表征速度慢

长文本容易出现语义漂移的问题

对分词和预处理语料要求高

4、词向量+sif 或者 usif 或者 TFIDF 权重平滑

上面介绍的词向量相加平均的方法缺点就是没有考虑到单词的重要性，所以等于句子里所有的词都是一样的权重，sif 和 usif 通过随机游走的原理来推算出加权公式。

优点：

词向量在大量语料中学到了语义

训练成本低，训练简单

考虑到单词的重要性，短文本效果还可以

缺点：

未考虑语序

句子表征速度慢

长文本容易出现语义漂移的问题

对分词和预处理语料要求高

Bert 这种预训练模型的句向量也不能直接用于短文本匹配

为什么 Bert 中的 CLS 在未 fine tune 时作为 sentence embedding 性能非常糟糕？196 赞同 ·0 评论回答 

5、Bert-flow

其做法为在 Bert 后面接一个 flow 模型，flow 模型的训练是无监督的，并且 Bert 的参数是不参与训练的，只有 flow 的参数被优化，而训练的目标则为最大化预先计算好的 Bert 的句向量的似然函数，将向量空间进行映射到高斯分布向量空间，将词向量分布调整。

https://arxiv.org/pdf/2011.05864.pdf arxiv.org/pdf/2011.05864.pdf

6、Bert-whitening

去除特征间的相关性和让所有特征具有相同的均值和方差。

https://arxiv.org/pdf/2103.15316.pdf arxiv.org/pdf/2103.15316.pdf

SinGaln：细说 Bert-whitening 的原理 71 赞同 · 5 评论文章 

7、SimCSE 和 ESimCSE

通过对比学习，进行自监督的方法，使用计算交叉熵为 loss 通过 softmax 分类来学习正负样本相似度。ESimCSE 是 SimCSE 升级版。

SimCSE 是通过 dropout 两个句子产生两个相似的正负样本进行对比学习，来学习到文本匹配之间的关系。

ESimCSE 解决了 SimCSE 遗留的两个问题：

1、SimCSE 通过 dropout 构建的正例对包含相同长度的信息（原因：Transformer 的 Position Embedding），会使模型倾向于认为相同或相似长度的句子在语义上更相似；

2、更大的 batch size 会导致 SimCSE 性能下降;

ESimCSE 构建正例对的方法：Word Repetition（单词重复）和 Momentum Contrast（动量对比学习）扩展负样本对。

SimCSE 论文：

https://arxiv.org/pdf/2104.08821.pdf arxiv.org/pdf/2104.08821.pdf

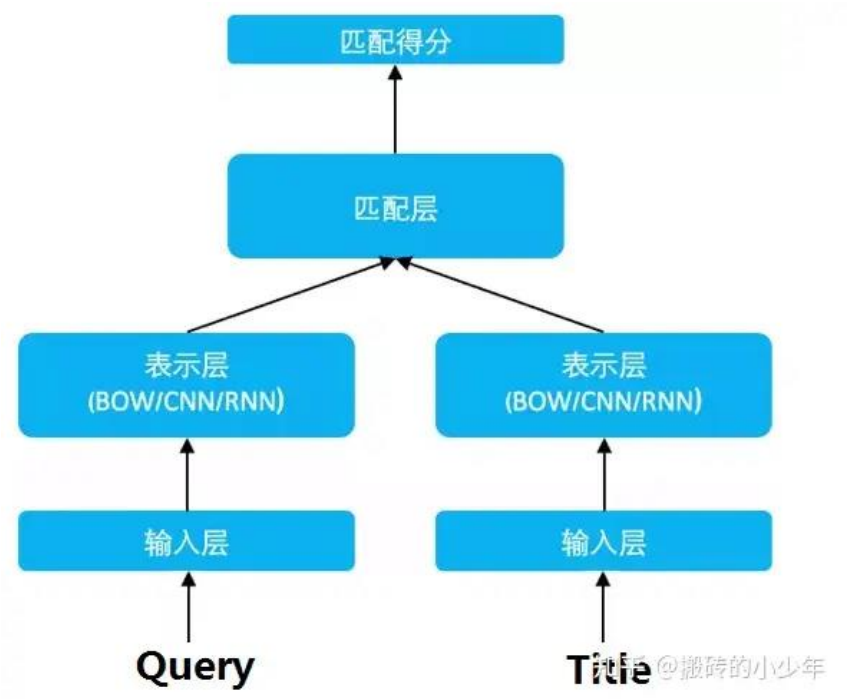
ESimCSE 论文：

https://arxiv.org/abs/2109.04380 arxiv.org/abs/2109.04380

2.2 有监督短文本匹配

1、表示型短文本匹配模型（经典双塔模型 DSSM）

双塔模型是很经典的文本匹配模型，多用于搜索、推荐、计算广告中召回部分，在工程上有很高的落地收益价值。



双塔模型结构原理是 query 和 title 在训练的时候分别从模型的两边输入，然后进入匹配层来判断是否是相似或者匹配，双塔模型属于表示型方法，所谓表示型模型就是最后两句 embedding 表征之后通过点积信息来匹配学习。

上图表示层的种类可以很多，比如普通的全连接层、LSTM 、CNN、Transformer、Bert 等，表示层权重共享。

在工业界使用中，一般是提前将 title 层表征完成的向量提前存入数据库，线上推理的时候模型只需要表征 query 然后通过 Faiss 高性能框架来进行相似度检索，大大提高检索性能，所以工业界中使用表示型模型多。

优点：

性能好，速度快

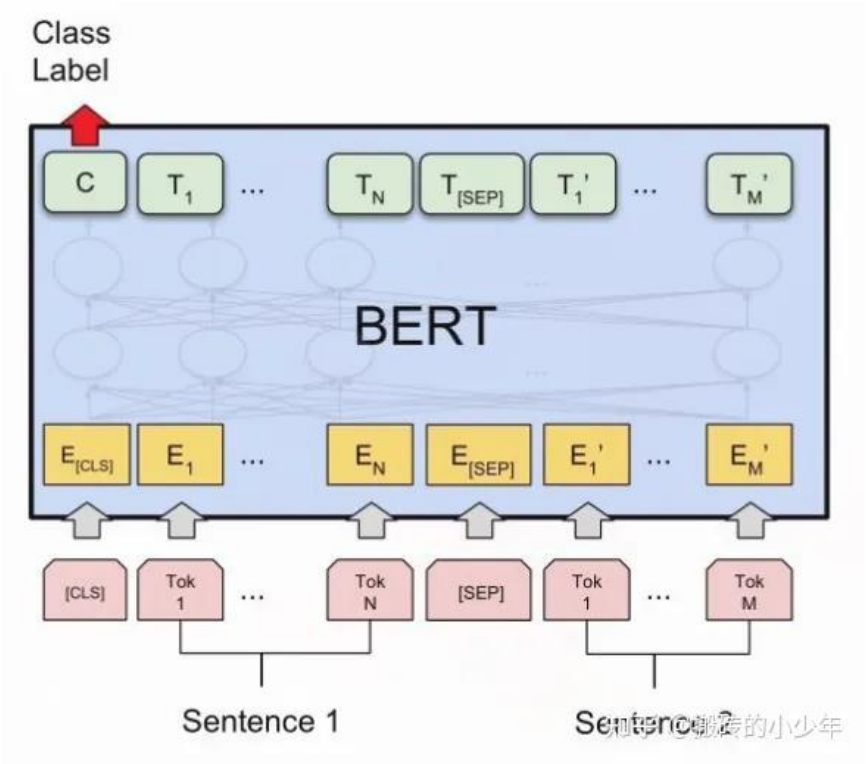
效果还不错，能学习到深层语义

缺点：

表示型模型在匹配层进行点积交互，点击交互信息太少会丢失一部分深层语义信息

2、交互型（单塔模型）

所谓单塔模型就是用一个模型来进行相似度匹配，这其实也可以看作是一个二分类问题，可以直接上 Bert 模型。



Bert 进行文本匹配在输入文本上可能需要做一些处理[CLS] query [SEP] title [SEP]来拼接两句话一起送入模型，这就是交互型模型，通过 Bert 学习到完整的两句话之间的强语义关系送入 12 层模型学习，不像表示型模型只有点积信息可以交互，效果上肯定是交互型模型更好，但是在工业界应用上可能主要用在语义精排上，因为每次 query 请求之后都要和所有数据库中的句子拼接进行拼接输入模型推理，速度太慢了，所以只能等召回一部分数据之后再进行精确语义的排序，无法在大数据量上使用。

优点：

效果好，能学习到深层语义

缺点：

大数据量下速度太慢

3、总结

监督短文本匹配和无监督短文本匹配在工业界都有应用，一般情况下会先尝试无监督短文本匹配的方法，如果效果达不到之后再去尝试监督短文本匹配，这也很正常毕竟标注数据成本挺高的，随着对比学习的发展，无监督短文本匹配的效果开始逐渐上升，能够更好的达到工业界应

用目的产生收益。