



# RAG研究(1.8)

## 背景介绍

尽管大模型的能力令人深刻，但仍然会存在一些问题。这些大模型可能会产生误导性的“幻觉”，依赖的信息过时，处理特定知识时效性不高，对专业领域的知识缺乏，同时在推理能力上也有所欠缺等。

幻觉：模型依赖于token by token的形式输出回答，难免会产生“一本正经的胡说八道”的情况

记忆力有限：实验证明，大模型在上下文对话的过程中容易丢失以前的知识。也就是说，模型的长时间记忆能力有限

时效性：模型输出结果过时、知识需要及时更新等

安全和隐私问题：模型的存取过程十分复杂，存在安全隐患。

与用户无关的建模方式：LLM还不具备针对特定群体建模的能力。出于这个目的，如何创造“轻量化”“定制化”产品成为主流。

新型、私有化的知识

- 查询增强
- 现存资料和备选资料
- 实验室--zhixun
- 徐雪乾-AI工程师（智能客服）

## RAG特点

- 可解释性/透明性：通过引用检索过程的检索词，用户可以验证答案的准确性，增加对模型输出的信任
- 定制能力。RAG通过索引相关的文本语料库，可以根据不同的领域定制模型，为特定领域提供知识支持。
- 可扩展性。它可以处理大规模的数据集，而不需要更新所有的参数和创建训练集，使其在节约成本的同时更有效。
- 结果更值得信赖。RAG 从最新的数据中选择deterministic结果，而微调模型在处理动态数据时可能出现幻觉和不准确，缺乏透明度和可信度
- 可更新性：和可扩展性类似，检索增强模型可以无需重新训练而只需更新或替换其依托的语料库实现模型的时时更新。
- 私有知识的安全：随着人工智能的发展，数据安全成为了企业关注的重点。对于企业而言，将长期累积的独有的知识库、敏感的经营数据、合同文件等机密信息上传到互联网上的大型模型可能会带来安全风险。在这种背景下，RAG技术提供了一种有效的解决方案。

## RAG技术

### Retrieval-Augmented Generation for Large Language Models: A Survey

**Yunfan Gao<sup>1</sup>, Yun Xiong<sup>2</sup>, Xinyu Gao<sup>2</sup>, Kangxiang Jia<sup>2</sup>, Jinliu Pan<sup>2</sup>, Yuxi Bi<sup>3</sup>, Yi Dai<sup>1</sup>, Jiawei Sun<sup>1</sup> and Haofen Wang<sup>1,3 \*</sup>**

<sup>1</sup> Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University

<sup>2</sup> Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University

<sup>3</sup> College of Design and Innovation, Tongji University

gaoyunfan1602@gmail.com

- 纯参数化语言模型(LLM)将其从大量语料库中获得的世界知识存储在模型的参数中。然而，这些模型也有其局限性。首先，培训语料库中的所有知识都很难保留，特别是那些不太常见的、具体的知识。其次，由于模型参数不能动态更新，随着时间的推移，参数知识很容易变得过时。最后，参数的扩展导致了训练和推理成本增加。为了解决纯参数化模型的局限性，语言模型可以采用半参数化的方法，将非参数化的语料库与参数化模型集成在一起。这种方法被称为检索增强生成(RAG)。
- RAG范式：预训练的检索器+预训练的 seq2seq 模型(生成器)结合起来，并进行端到端的微调，以更易于解释和模块化的方式捕获知识（通俗理解：RAG为模型提供教科书，允许它基于特定查询检索信息）
  - 检索：利用编码模型检索基于问题的相关文档，如 BM25、DPR、ColBERT 和类似的方法
  - 生成：使用检索到的上下文作为条件，系统生成文本。

- RAG流程：RAG 动态地从外部知识源检索信息，使用检索到的数据作为引用来组织答案。这大大提高了反应的准确性和相关性，有效地解决了存在于 LLM 中的幻觉问题

(问题：针对同一个输入语句，Chatbot A(比如chatgpt)能不能通过RAG检索Chatbot B(比如智谱清言)的知识，从而集成多个chatbot的输出结果)

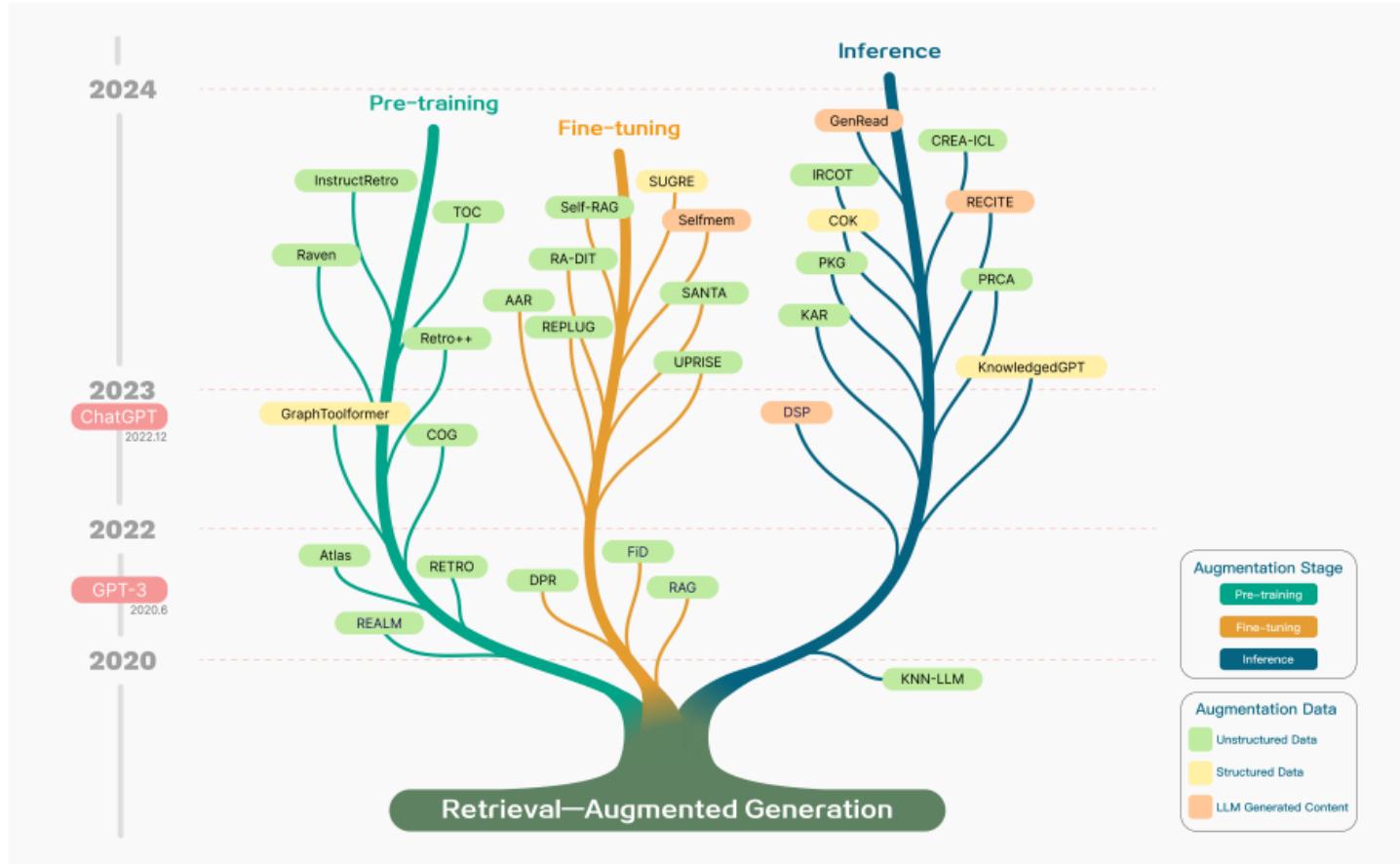


Figure 1: A timeline of existing RAG research. The timeline was established mainly according to the release date.

## 基于微调的RAG

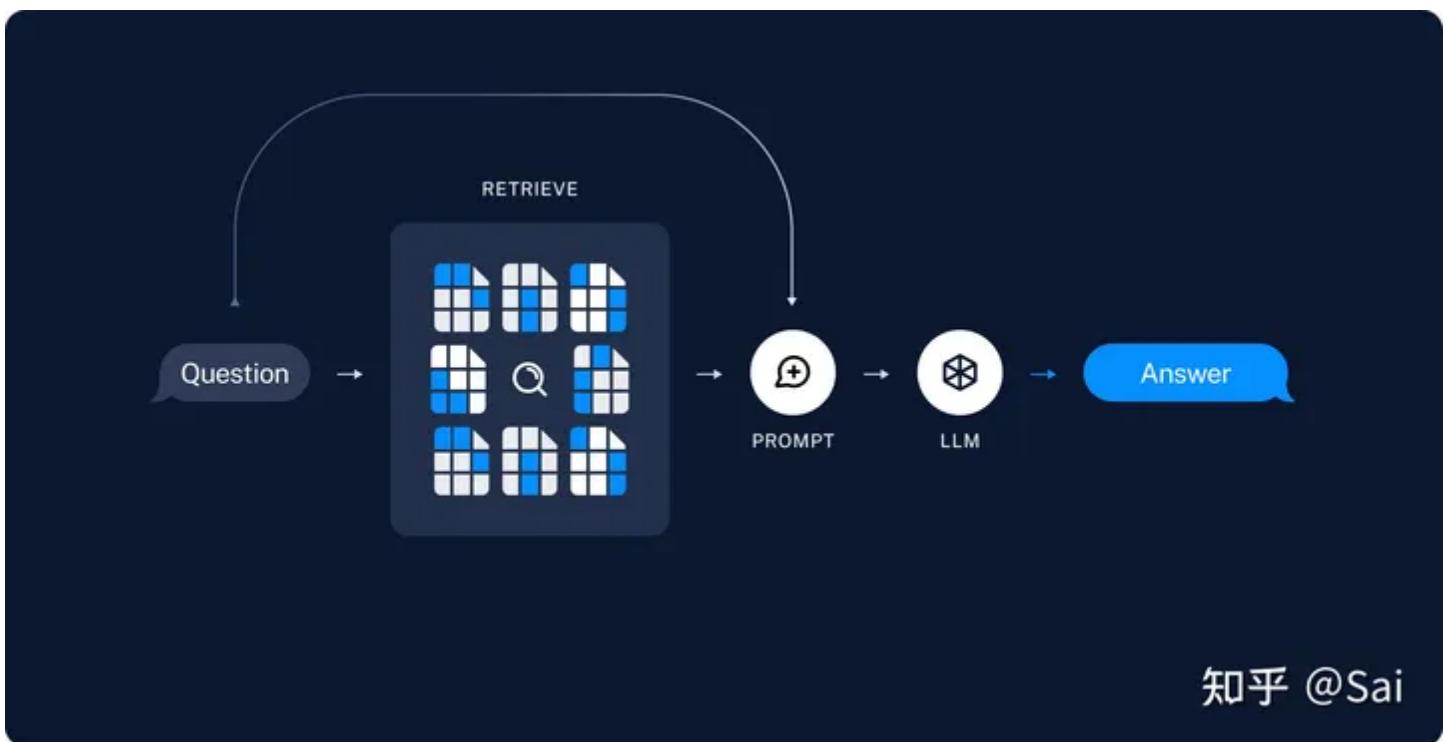
微调特别适合于强调基础模型中的现有知识，修改或定制模型的输出，以及用复杂的指令指导模型。然而，微调不适合于向模型中添加新知识，也不适合于需要对新用例进行快速迭代的场景。

结论：RAG 和微调相互补充，从而增强模型的能力。在某些情况下，将这两种技术结合起来可以获得最佳的模型性能。RAG 优化和微调的整个过程可能需要迭代之后才能达到满意的结果。

	RAG	Fine-tuning
特点	类似于为模型提供教科书，允许它根据特定查询检索信息	类似于让学生通过广泛的学习内化知识，特别适合于强调基础模型中的现有知识，修改或自定义模型的输出，以及为模型提供复杂的指令
缺点	不适合教授模型理解广泛的领域或学习新的语言、格式或风格	不适合将新知识纳入模型，也不适合需要快速迭代新用例的情况。

图：RAG vs. Fine-tuning

## RAG过程



图：RAG分为检索和生成阶段。

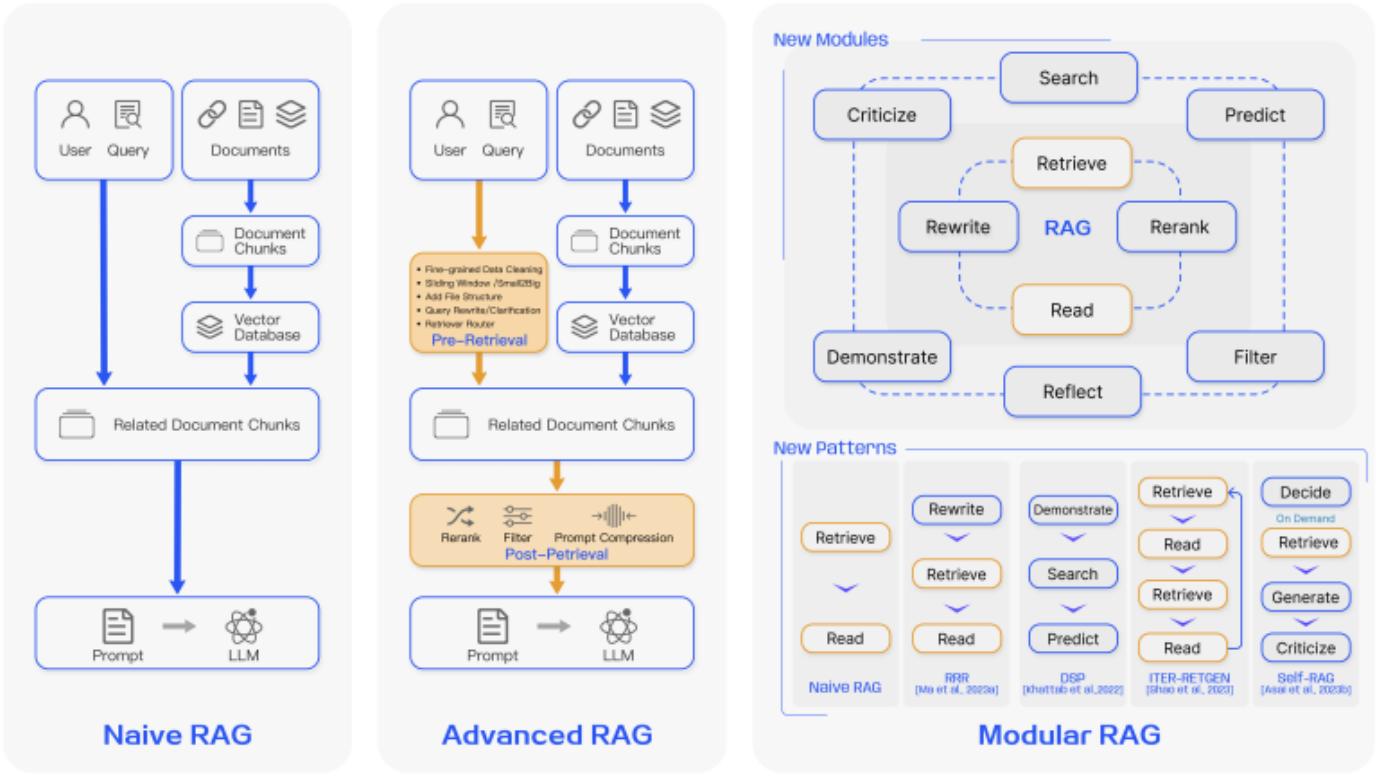


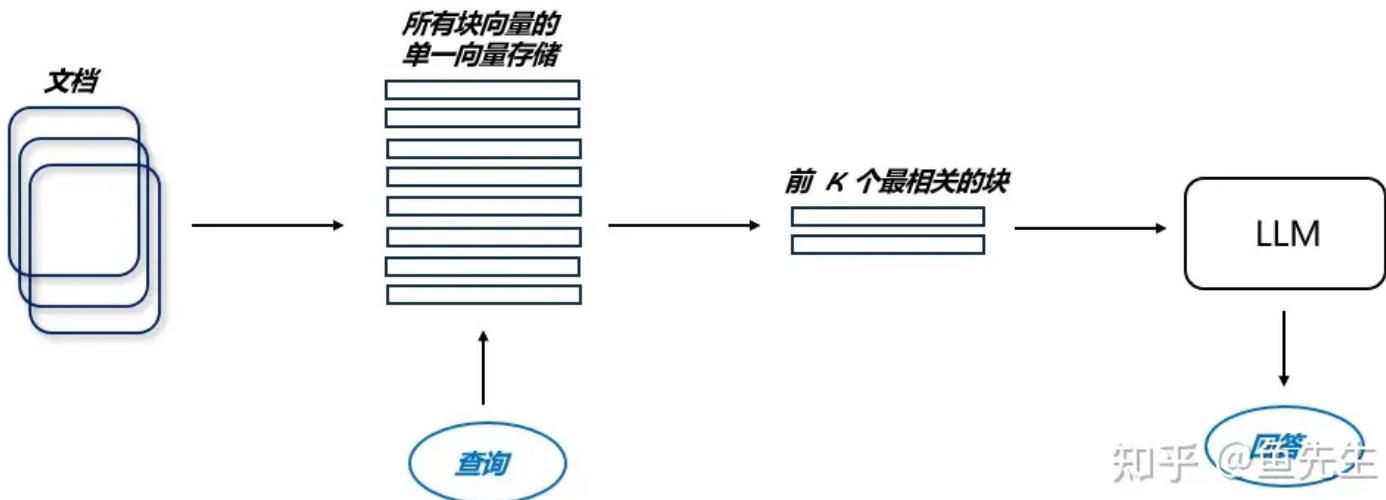
Figure 3: Comparison between the three paradigms of RAG

图：RAG 三种范式的比较

## Naive RAG

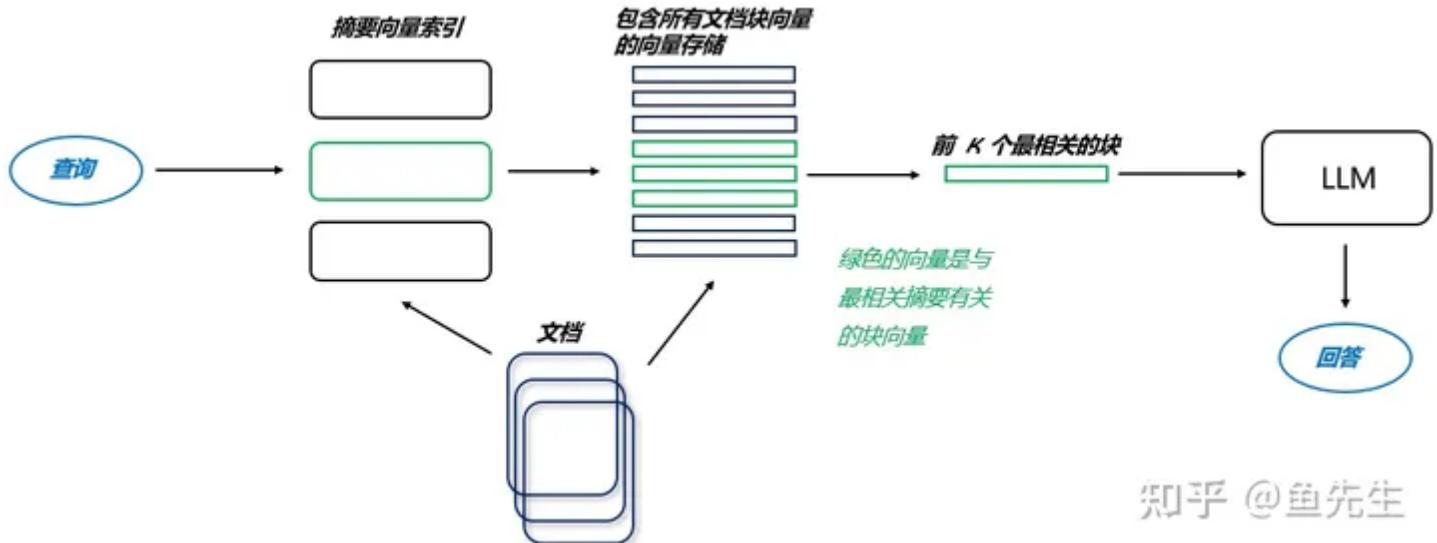
流程（6阶段）：1索引(外部数据源) --> 2数据索引（清洗&预处理）--> 3分块（chunking，为模型提供固定的上下文区间）--> 4嵌入（得到向量化形式）--> 5提取(retrieve，给定用户输入，使用编码器得到输入向量，然后和上一步的所有chunk计算相似度，取topK作为增强信息）--> 6生成（此时用户输入+文档chunk作为Prompt给到模型，用于比如multi-round dialogue或者single-round dialogue任务）

缺点：不能很好地结合上下文和外部数据源的检索内容

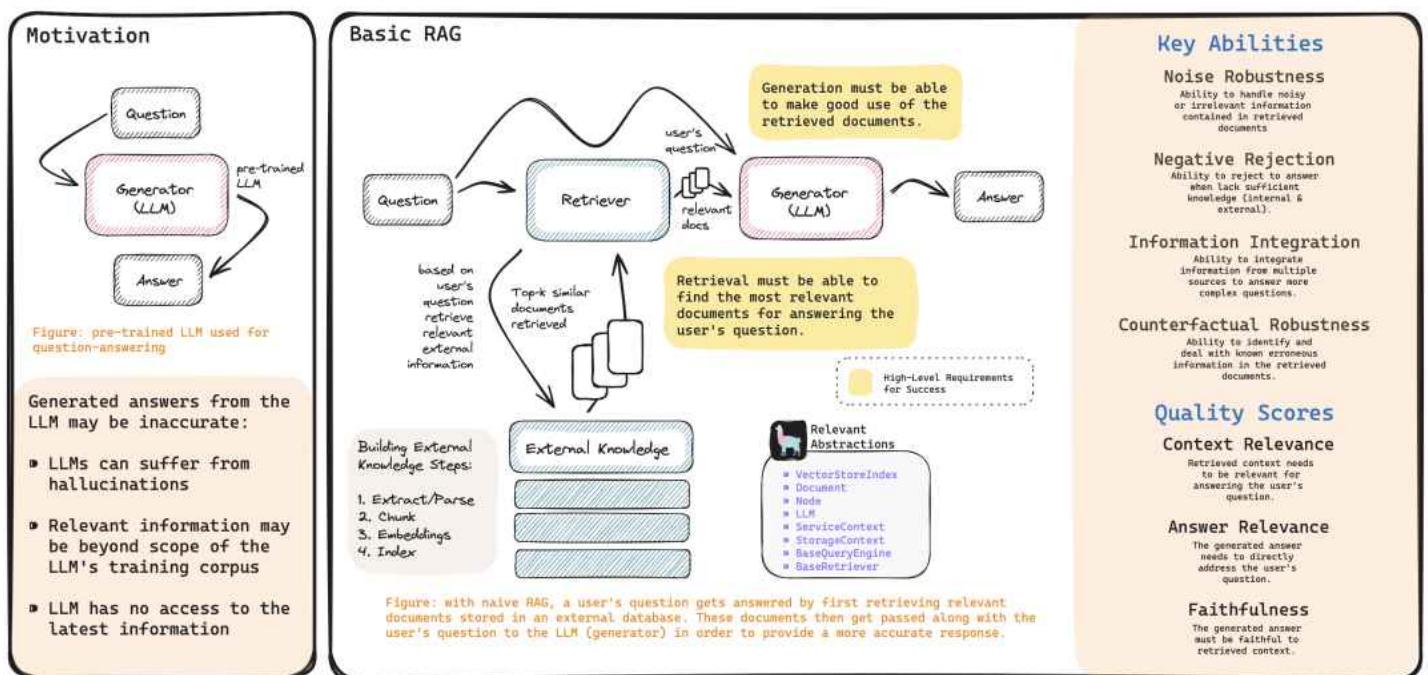


图：查询内容会首先进行向量化。类似地，尽管索引是根据向量而不是具体的块来进行检索的，但最终我们还是以块的形式展现结果，因为获取这些块相对简单。

## 改进版：



图：对于大型数据库，一个有效的方法是创建两个索引：一个包含摘要，另一个包含所有的文档块，并通过两步搜索，先利用摘要筛选相关文档，再在这些相关文档中进行具体搜索。



图：Naive RAG的概览图，包括了其主要特点、评估方法等。

目前主流的向量化工具：

模型名称	简介	地址
ChatGPT-Embedding	ChatGPT-Embedding由OpenAI公司提供，以接口形式调用。	<a href="https://platform.openai.com/docs/guides/embeddings/what-are-embeddings">https://platform.openai.com/docs/guides/embeddings/what-are-embeddings</a>

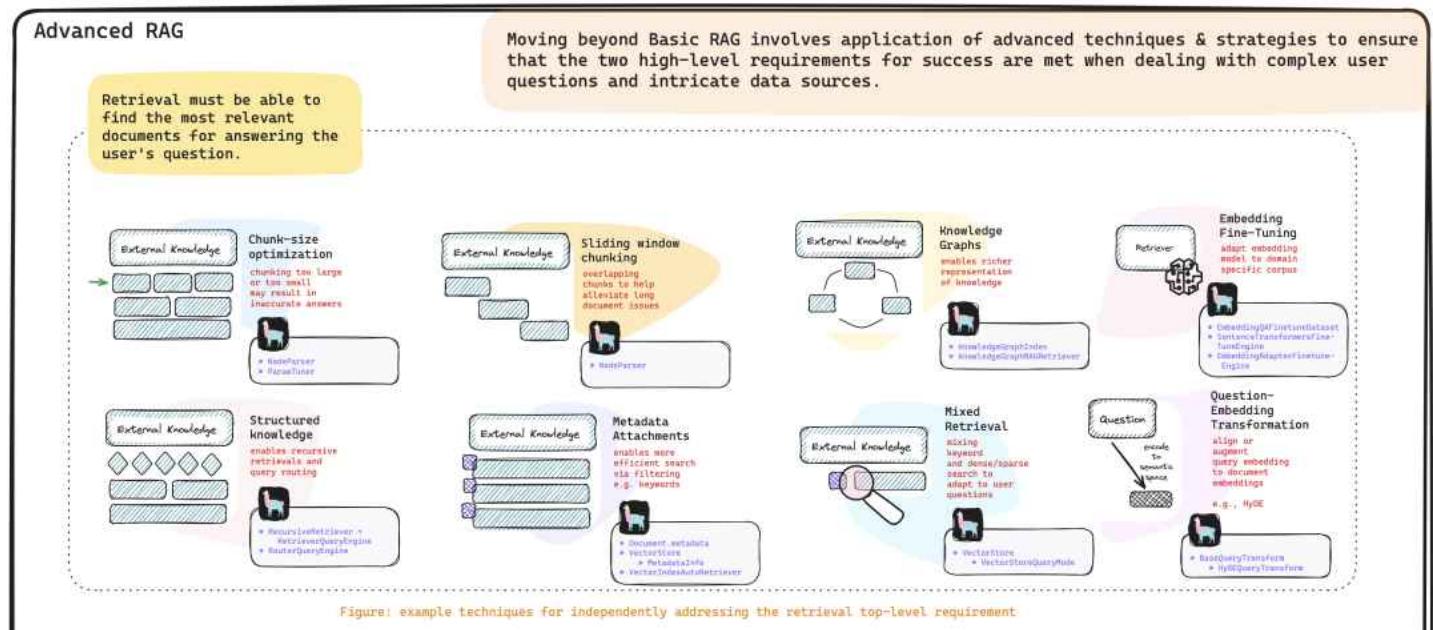
ERNIE-Embedding V1	ERNIE-Embedding V1由百度公司提供，依赖于文心大模型能力，以接口形式调用。	<a href="https://cloud.baidu.com/doc/WENXINWORKSHOP/s/alj562vvu">https://cloud.baidu.com/doc/WENXINWORKSHOP/s/alj562vvu</a>
M3E	M3E是一款功能强大的开源Embedding模型，包含m3e-small、m3e-base、m3e-large等多个版本，支持微调和本地部署。	<a href="https://huggingface.co/moka-ai/m3e-base">https://huggingface.co/moka-ai/m3e-base</a>
BGE	BGE由北京智源人工智能研究院发布，同样是一款功能强大的开源Embedding模型，包含了支持中文和英文的多个版本，同样支持微调和本地部署。	<a href="https://huggingface.co/BAAI/bge-base-en-v1.5">https://huggingface.co/BAAI/bge-base-en-v1.5</a>

## Advanced RAG

优化上面提到的6阶段

优化2阶段：优化数据粒度(及时清除无关信息，过时信息) or 优化数据索引结构(查询效率)/引入图结构存储检索信息 or 加入元数据(使用chapter/section标注数据的相对位置,提高retrieval效率)

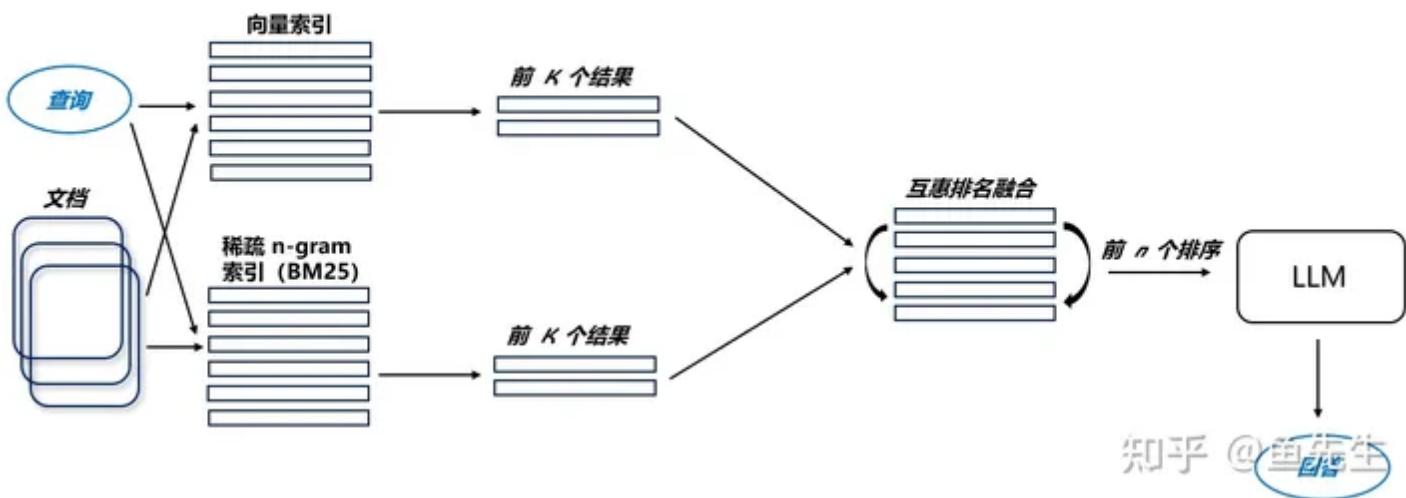
优化6阶段：压缩prompt(清除冗余信息，保证token长度合法)



图：Advanced RAG的概览图，它是Naive RAG的优化版本，加入了滑动窗口分块、知识关系的图建模、引入prompt的文档嵌入等方法。

## RAG pipeline优化

Hybrid search: keyword-search + semantic search + vector search



图：混合检索示意图

**Recursive search(query engine):** 递归检索需要在初始检索阶段获取较小的文档块，以捕获关键语义。在此过程的后期阶段，将向语言模型(LM)提供具有更多上下文信息的最大块。这种两步检索方法有助于在效率和上下文丰富的响应之间取得平衡。

## 模块化的RAG

搜索模块

记忆模块

额外生成模块: 额外的生成模块利用 LLM 生成所需的上下文。与直接检索相比，LLM 生成的内容更可能包含相关信息

与下游任务相关的模块： UPRISE自动从预先构建的数据池中检索给定zero-shot输入的提示，增强了任务和模型之间的通用性。PROMPTAGATOR利用 LLM 作为few-shot查询生成器，并根据生成的数据创建特定于任务的检索器。利用 LLM 的泛化能力，PROMPTAGator 只需几个示例就可以创建特定于任务的端到端检索程序。

对齐模块： 查询和文本之间的对齐一直是影响 RAG 有效性的关键问题。PRCA使用强化学习训练由 LLM 奖励驱动的上下文适配器，位于检索器和生成器之间。它通过在标记的自回归策略的强化学习阶段最大化奖励来优化检索到的信息。AAR提出了一个通用的插件，学习从已知来源 LLM 的 LM 偏好，以协助未知或非共同微调 non-co-finetuned LLM。RRR设计了一个基于强化学习重写查询的模块，使查询与语料库中的文档保持一致。

（问题：对齐模块和任务模块的区别？？）

验证模块： 保证输出结果有效性

## 术语名词



## Retrieval system:

**Hallucination:** 大模型幻觉，特质大模型出现“一本正经的胡说八道”的情况(本质原因是模型根据token by token进行输出的，难免会丢失一部分语义的连贯一致性)

**Parametric knowledge:** 是通过训练LLM获得的，并存储在神经网络权重中，代表模型对训练数据的理解和概括，形成生成响应的基础。

**Non-parametric knowledge:** 存在于矢量数据库等外部知识源中，不直接编码到模型中，而是作为可更新的补充信息处理。

## Vector database

## Vector similarity search

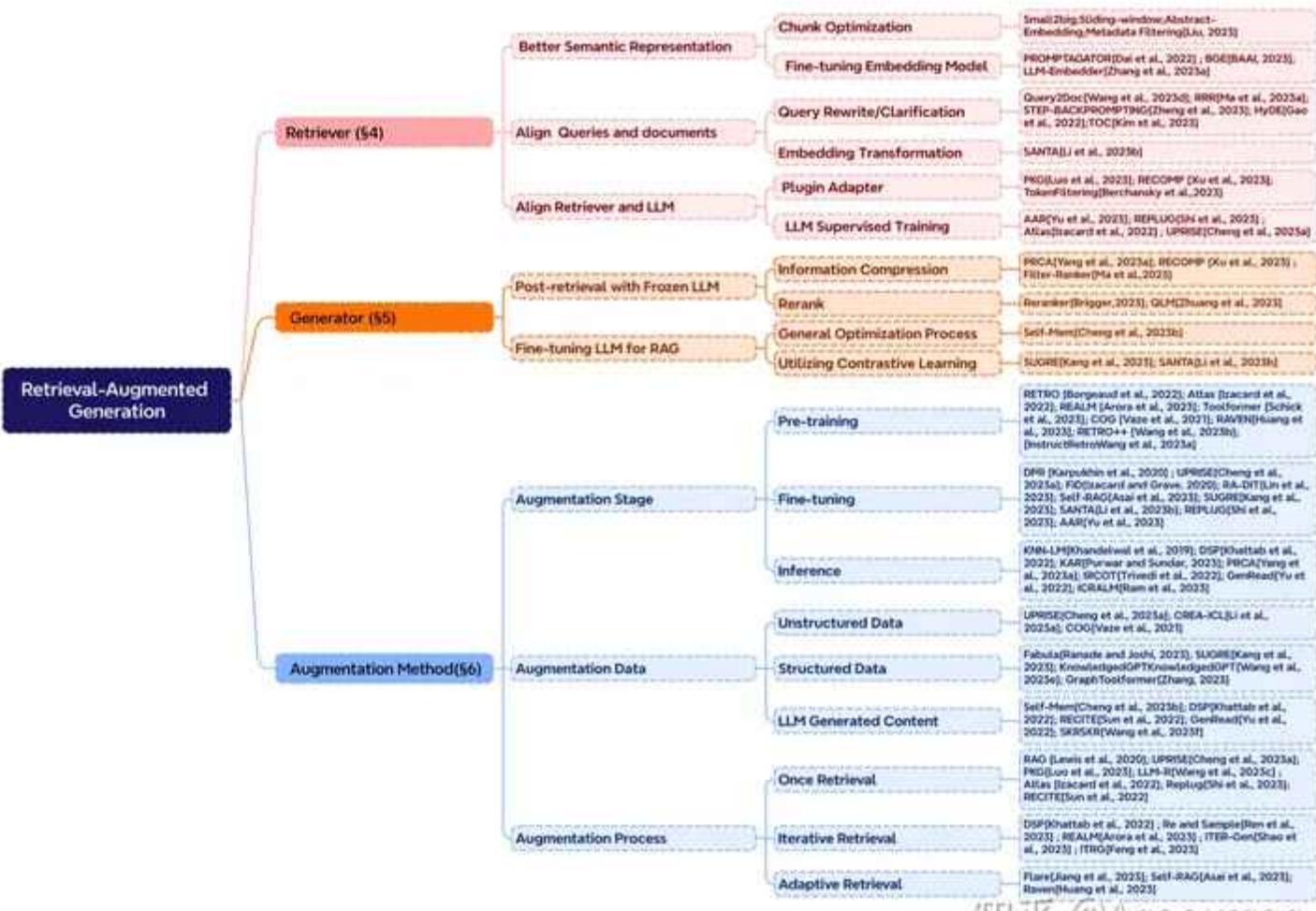


Figure 4: Taxonomy of RAG's Core Components

图：RAG核心组件

# RAG系统

完整的RAG应用流程主要包含两个阶段：

- 数据准备阶段：数据提取——>文本分割——>向量化 (embedding) ——>数据入库
- 应用阶段：用户提问——>数据检索（召回）——>注入Prompt——>LLM生成答案



<https://zhuanlan.zhihu.com/p/674093669>

## self-RAG：一种新的LLM+RAG框架

现在LLM+RAG（检索增强）的技术方案已成为LLM在众多应用场景的主流。在LLM+RAG技术框架下，存在的问题/技术难点主要有：1) 如何保证检索内容是有效…

<https://selfrag.github.io/>

## Self-RAG: Learning to Retrieve, Generate and Critique through Self-Reflection

Self-RAG: Learning to Retrieve, Generate and Critique through Self-Reflection.

<https://github.com/AkariAsai/self-rag>

## GitHub - AkariAsai/self-rag: This includes the original implementation of SELF-RAG: Learning to Retr

This includes the original implementation of SELF-RAG: Learning to Retrieve, Generate and Critique through self-reflectio…

## Self-RAG

### Self-RAG与普通RAG优势对比

自适应段落检索：通过这种方式，LLM可以继续检索上下文，直到找到所有相关的上下文（当然是在上下文窗口内）；

更相关的检索：嵌入模型通常并不擅长检索相关上下文。Self-RAG可能通过[relevant]的特殊token来解决这一问题；

击败其他同级别的模型：Self-RAG在许多任务中也出人意料地击败了ChatGPT。

不会改变基本的LM：微调和RLHF很容易导致模型产生偏差。Self-RAG似乎通过添加特殊的token来解决这个问题，并在其他方面保持文本生成不变。

```
from vllm import LLM, SamplingParams

model = LLM("selfrag/selfrag_llama2_7b",
download_dir="/gscratch/h2lab/akari/model_cache", dtype="half")
sampling_params = SamplingParams(temperature=0.0, top_p=1.0, max_tokens=100,
skip_special_tokens=False)

def format_prompt(input, paragraph=None):
    prompt = "### Instruction:\n{0}\n\n### Response:\n".format(input)
    if paragraph is not None:
        prompt += "[Retrieval]<paragraph>{0}</paragraph>".format(paragraph)
    return prompt

query_1 = "Leave odd one out: twitter, instagram, whatsapp."
query_2 = "Can you tell me the difference between llamas and alpacas?"
queries = [query_1, query_2]
```

```
# for a query that doesn't require retrieval
preds = model.generate([format_prompt(query) for query in queries],
sampling_params)
for pred in preds:
    print("Model prediction: {}".format(pred.outputs[0].text))
```

Self-RAG 在第一个查询不需要检索时开始生成不需要检索的响应。另一方面，第二个是 Self-RAG 输出 [ Retrieve ] 令牌，因为这个问题需要更细粒度的事实基础。对于需要事实基础的查询，可以插入一个段落。Self-RAG 可以在生成段落时随时检索和插入段落，并且只要段落被上下文标记特殊标记包围，就可以识别段落。

## 案例：QA助手

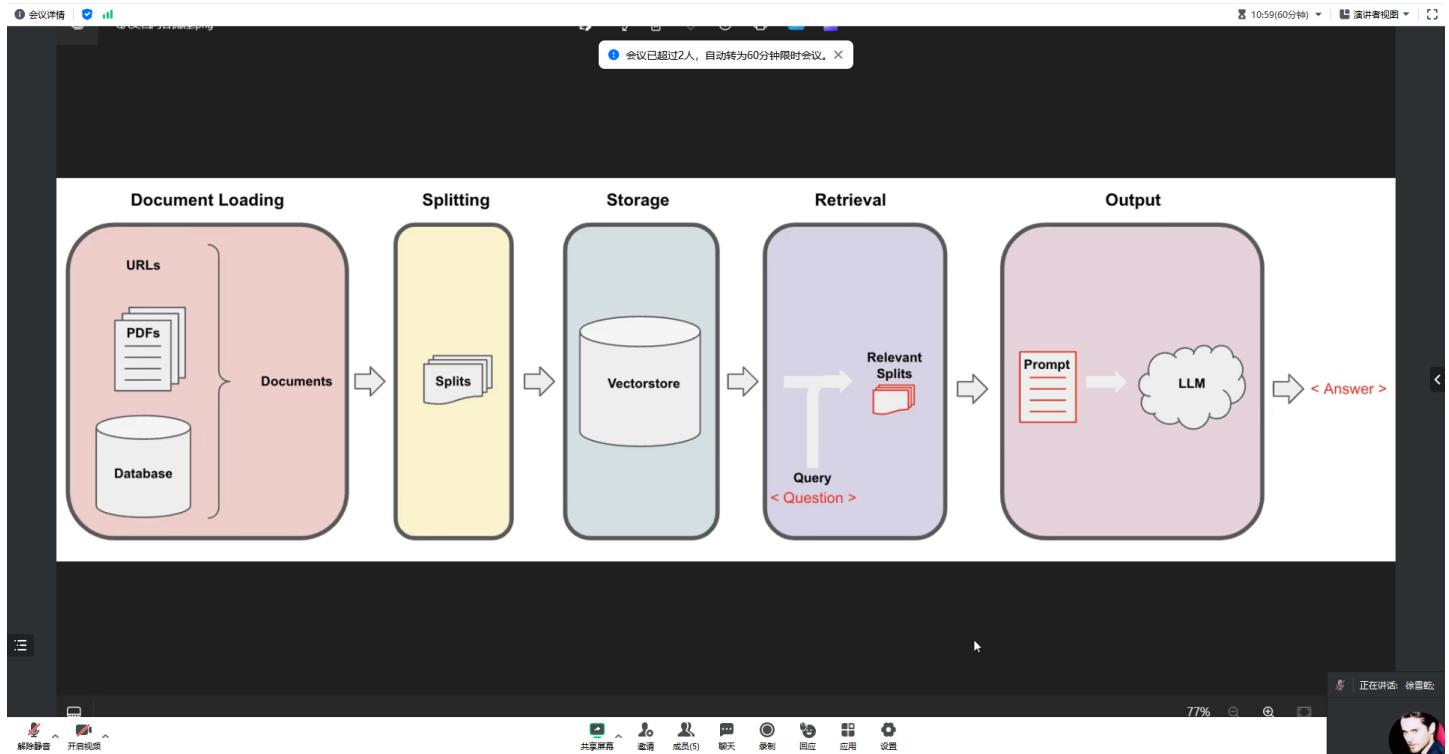


“Prompt的设计只有方法、没有语法”

### Pazhi

RAG技术的专业知识库、文档检索

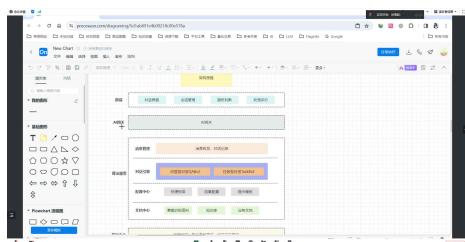
大模型+知识库问答+(对接企业内部的查询)



图：1文档加载：结构化&非结构化数据(多模态)。2文档片段：计算文档-用户输入相似度(字数分段、问答对分段)。3向量存储。4检索：topN(<Question>)检索结果是文本向量编码)。5输出：提示工程（prompt+文本+大模型=<Answer>）

国外：RAG--维基百科检索--斯坦福

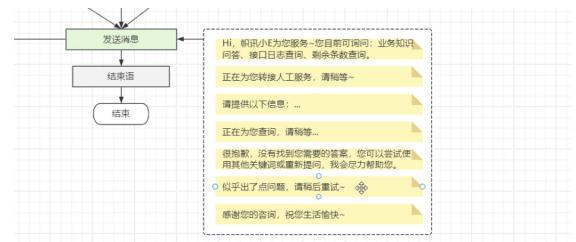
我们：知识手册



图：界面、对话引擎(开放型对话引擎/chatgpt、问答型对话)、文档中心、大数据存储引擎



图：任务型对话(用于意图识别)难度较大，生成sql代码



图：调用chatgpt接口实现智能客服



Q: 问答型对话和任务型对话和开放型对话的区别是？

问答型是指根据内部知识回答，任务型的是调用外部工具或API，开放型的是大模型的能力

Q: 为了达到符合时效的、准确的回答，大模型可能需要检索大量文档，这会不会得不偿失？以及如何保证检索内容的客观性、准确性？

检索方式上有没有优化算法？

**Q: 针对一些“网红问题”，大模型怎么做出道德评判和决策的？不公正的回答这可能会威胁到企业安全**

**Q: 两个接口实现的意图识别？**

函数名的分类

**Q: 关键挑战？**

意图识别

**Q: 向量搜索优化的基本方式？**

**Q: 有没有可能，使用prompt engineering从多个LLMs获取答案，然后传递到我们的LLMs，从而实现生成质量的提高？**

## 测试

U: 你好  
A: Hi, 帐讯小E为您服务~  
您可询问以下相关内容：  
1. 接口日志查询  
2. 剩余条数查询  
3. 业务知识问答  
4. 转接人工服务

U: 原神是一款什么类型的游戏  
A: 很抱歉，没有找到您需要的答案，您可以尝试使用其他关键词或重新提问，我会尽力帮助您。

U: 通用短信撤回方式  
A: 很抱歉，我不知道通用短信的撤回方式。

U: 多签名机制  
A: 多签名机制是指给某个企业配置了多个可以使用的企业签名。通过多签名机制，企业用户可以在发送短信时选择不同的签名，以适应不同的业务场景。管理员可以在所需要的签名，并将所需签名对应的“签名编号”传输到接口参数中的“signcode”中，从而实现多签名的功能。

U: 13910639260帮忙看下有接口报错么  
A: 请提供以下信息：企业编号，查询时间

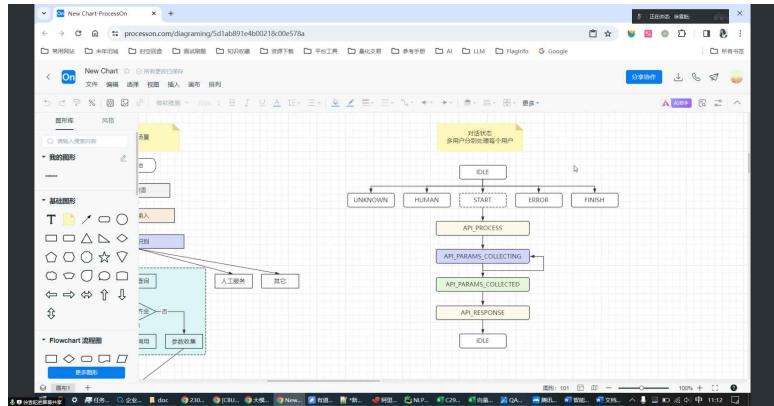
U: 123456 11.22  
A: 正在为您查询，请稍等...  
A: API1: 接口报错记录

U: 好的，感谢  
A: 感谢您的咨询，祝您生活愉快~

图：意图识别功能演示

图：问题与问题的匹配度、问题与答案的匹配度

图：整体准确率93%



图：基于上下文判断用户下一次的意图。多轮对话问题、多人会话Session问题。历史消息---对话状态管理。

## RAG开源工具

<https://www.marktechpost.com/2024/01/10/8-open-source-tools-for-retrieval-augmented-generation-rag-implementations/>

### 8 Open-Source Tools for Retrieval-Augmented Generation (RAG) Implementation

8 Open-Source Tools for Retrieval-Augmented Generation (RAG) Implementation

## 套路

1. 基于文档整理答案
2. 计算相似度
3. 文本---向量：向量数据库

**数据库：向量存储&向量检索**

**大模型：百川、chatgpt**

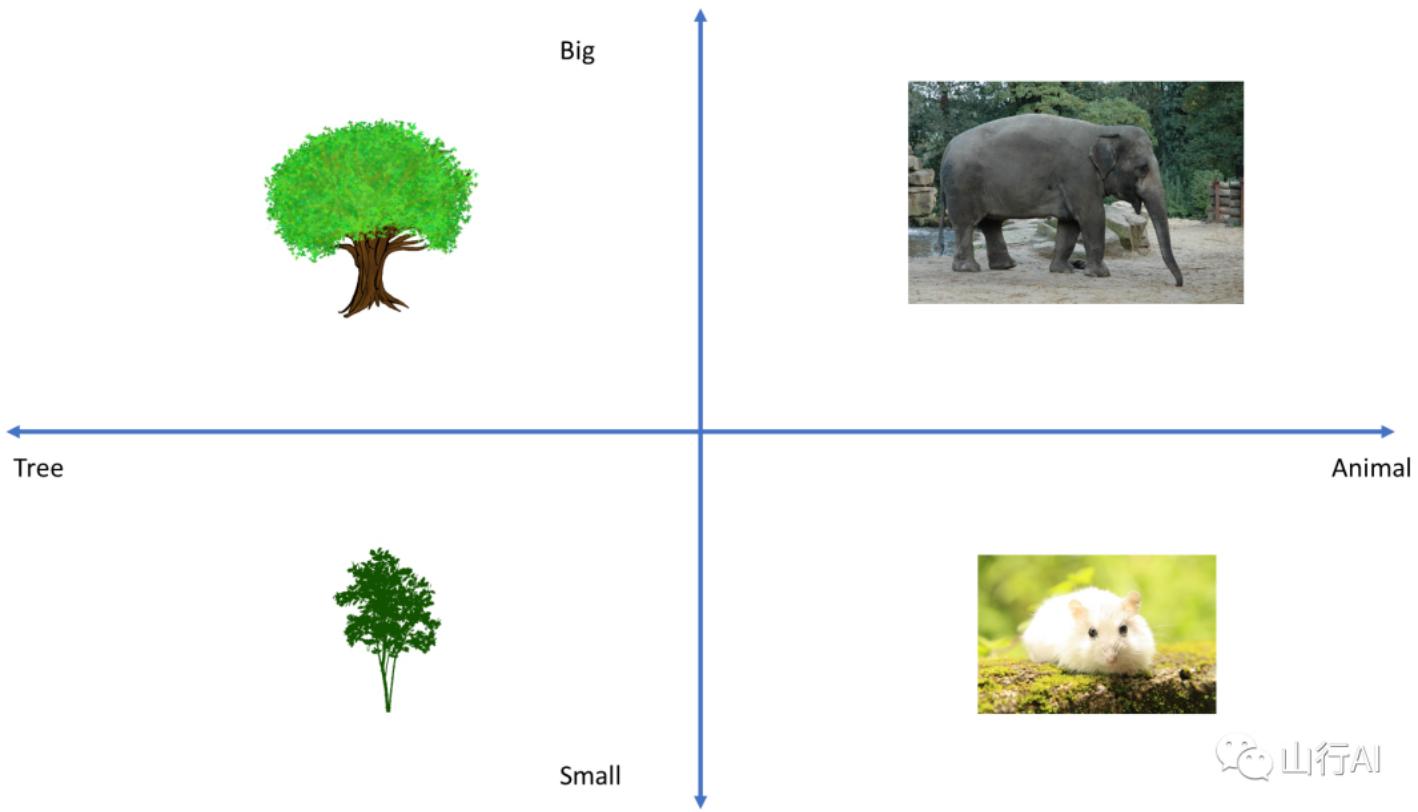
**向量化：**把输入特征转化到维度，可以跟one-hot编码对比。比如，数据是一组图像，向量化形式就是一个元组：大小和类型，分别用来描述对象的不同属性，大象是动物并且是大的，香菜是植物并且是小的。这只是一个概念性的例子，对于更多（数百）属性，我们只需要增加n个维度用来区别。

### 知识问答

1. 文档处理和数据处理
2. 文档分割
3. 问题与问题之间
4. 问题与答案的匹配度
5. 向量存储与向量检索
6. 开源的和非开源的

### 任务型对话

1. 意图识别：增加训练语料或提升语料
2. 对话管理
3. 历史消息
4. 工程实现：权限、数据查询、消息收发…



你可以用单词或句子做同样的事情，而不是图片。注意在上面的例子中，向量化能够捕捉到语义表示，即它知道一个句子谈论一只鸟猛扑一只小花栗鼠应该在（小，动物）象限，而谈论昨天的风暴时一棵大树倒在路上的句子应该在（大，树）象限。实际上，有超过2个维度 - 对于句子转换器嵌入（句子或文本块通过BERT转换器，输出对应于384个维度的384个数字）。



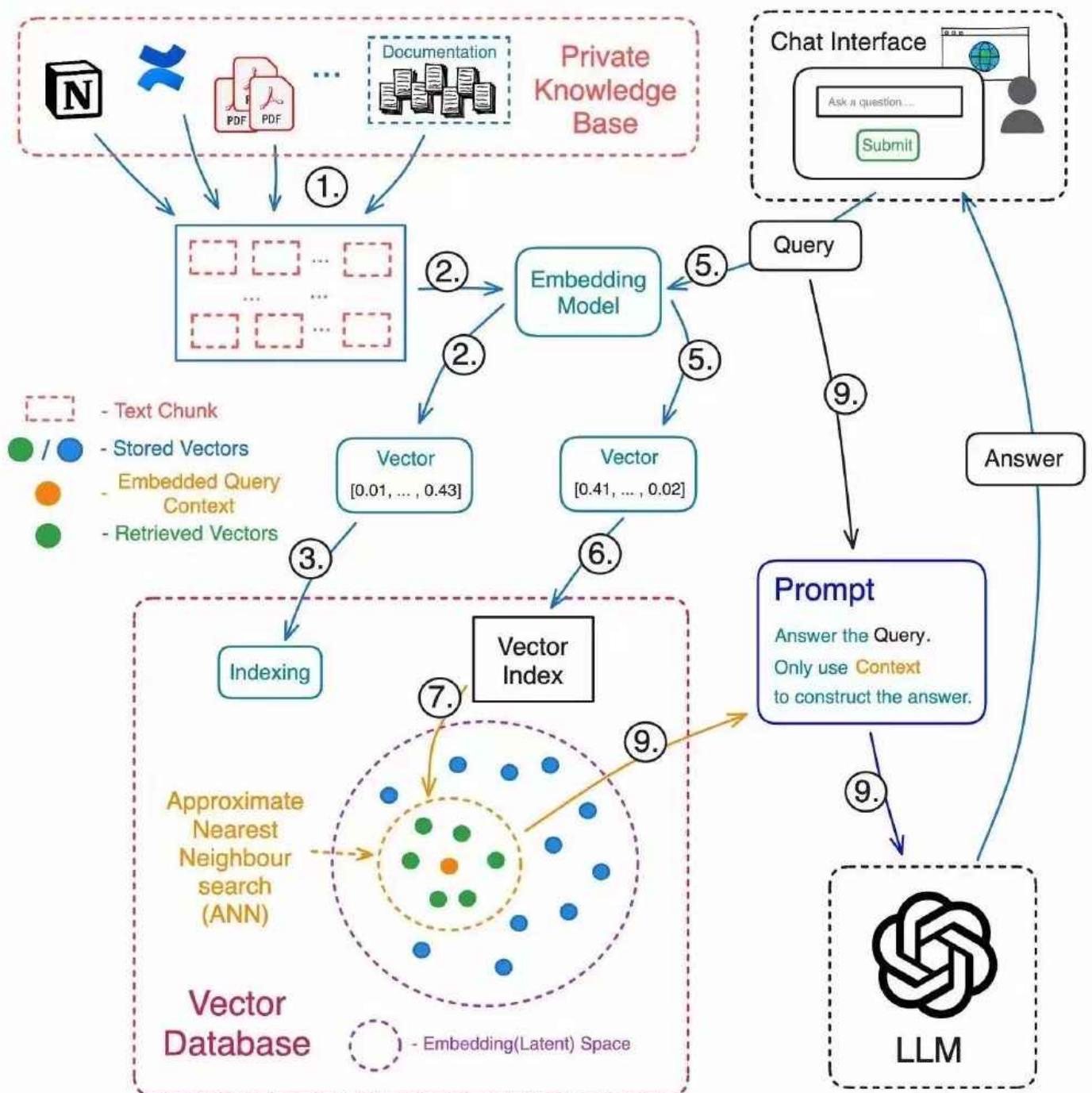
山行AI

Document	Vector Embeddings	Query	Most relevant Document
During yesterday's storm, a large tree fell on the road.	[1,0,0,0]	1	During yesterday's storm, a large tree fell on the road.
While I was driving on the road in Bandipur, a large elephant crossed the road.	[0,1,0,0]	2	I saw a bird swooping in on a baby chipmunk through my window
Bonsai are usually planted in tiny pots, they tend to dry up soon, so water your Bonsai regularly.	[0,0,1,0]	3	While I was driving on the road in Bandipur, a large elephant crossed the road.
I saw a bird swooping in on a baby chipmunk through my window	[0,0,0,1]	...	...



图：文档内容和向量嵌入结果（左）和用户查询结果（top3最相关的内容）

RAG原型：



图：RAG原型。左下方是向量数据库的聚类搜索，用来高效地存储query+document的嵌入向量

🌟 2和5为什么要共同只想一个嵌入模型？

3和6为什么要分别存储，3没有聚类？

相似度度量在哪里？

🎼 RAG不管多厉害，它还是基于LLM的。脱离了LLM，RAG会消失“人味”

💡 一些思考：通过了解RAG的架构和技术细节，我联想到一些名词：对于构建一个行业的垂直问答系统，我认为可以从以下几个现有工作吸取经验

专家系统：似乎是一个集成领域知识的知识库，可以充当比如语言领域的电子字典的角色。

专家积：MoE

行业大模型

微调

限制（可设置的参数）

## 最大块长度

。这些块应该是包含答案的典型（最小）大小的文本。这是因为你可能会问的问题可能在文档的多个位置有答案。例如，你可能会问“X公司从2015年到2020年的表现如何？”你可能有一个大的文档（或多个文档）包含关于公司在不同部分的文档中的年度表现的具体信息。你理想的目标是捕获所有包含这些信息的文档的不同部分，将它们链接在一起，并将其传递给一个大型语言模型，基于这些过滤和连接的文档块进行回答。

## 最大上下文长度

基本上是将各种块连接在一起的最大长度——留下一些空间用于问题本身和输出答案（记住，像ChatGPT这样的大型语言模型有一个严格的长度限制，包括所有的内容：问题，上下文和答案。）

## 相似度阈值

是比较问题与文档块的方式，找到最有可能包含答案的顶部块。余弦相似度是通常使用的度量，但你可能想要权重不同的度量。例如，包括一个关键词度量，以更多地权重包含某些关键词的上下文。例如，当你向大型语言模型提问以总结文档时，你可能希望权重包含“摘要”或“总结”这些词的上下文。

## 提高内容生成质量

从文档检索角度：高质量、有语义信息的各种模态的文档

从向量存储角度：查询高效、检索高效的向量database。我上面展示的架构是一个原型。为了效率和可扩展性，必须考虑各种方面，包括向量嵌入模型、文档数据库、提示、大型语言模型的选择等。

从相似度度量角度：有时，简单的余弦相似度是不够的。例如，如果许多文档包含关于同一主题的冲突信息，你可能希望根据这些文档中的元数据限制对某些文档的搜索。为此，除了相似性，你可以使用其他过滤度量。



相似性我认为是QA生成的关键和核心，承上连接了外部文档，启下连接了内容生成。好的相似度度量能决定生成质量的好坏，以及用户体验。基于这个目的，我调研了目前的主流方案：

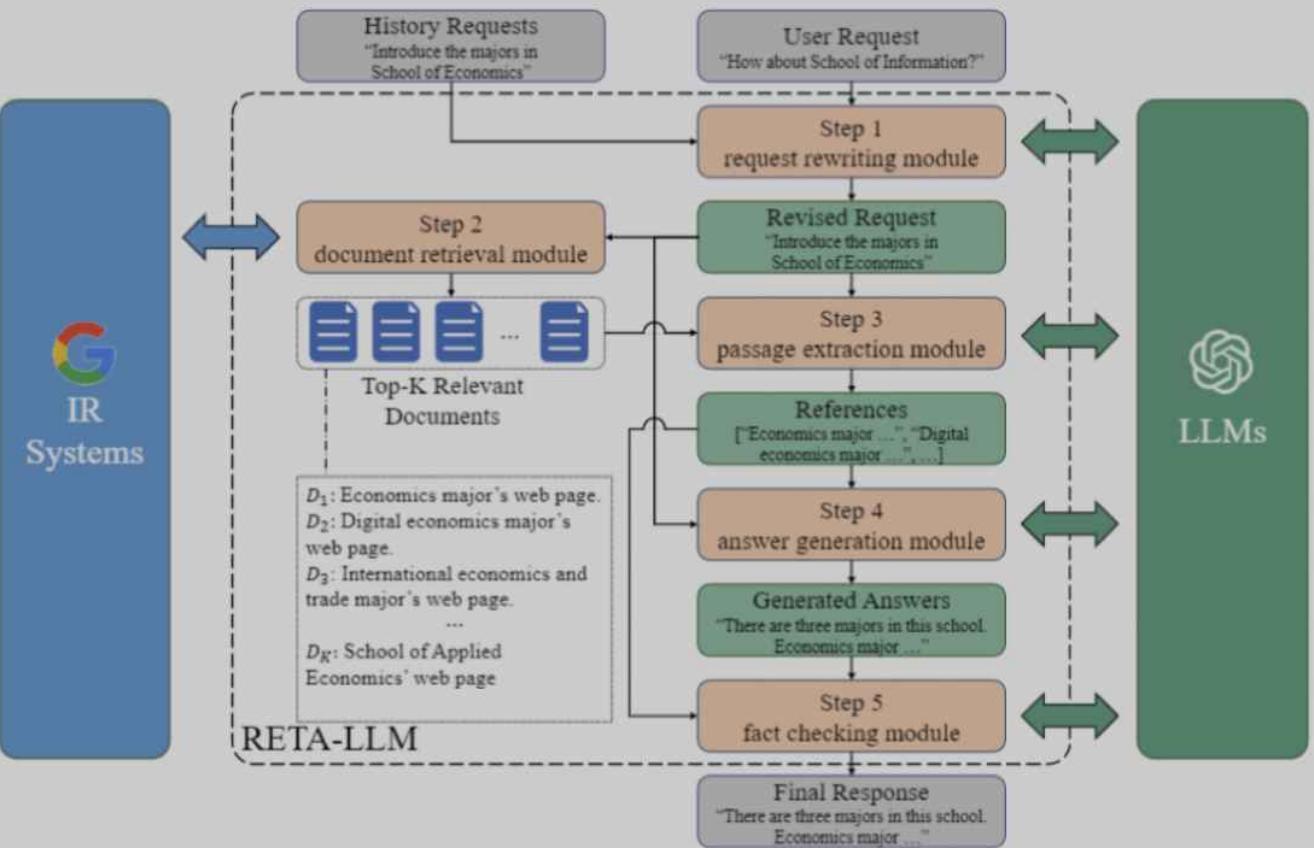


Figure 1: RETA-LLM framework.

检索模型=图书馆员

生成模型=作家

## 文档--有效内容

WebBrain=Wikipedia+Reference=高质量语料库

WebBrain能够利用检索到的相关文档片段，辅助生成高质量的富含知识的聚合文档，标注出相关引用，从而显著提升用户获取信息的效率，并保证生成内容的事实性和可信性

## 有效内容--嵌入向量

传统DB：查询结构化数据

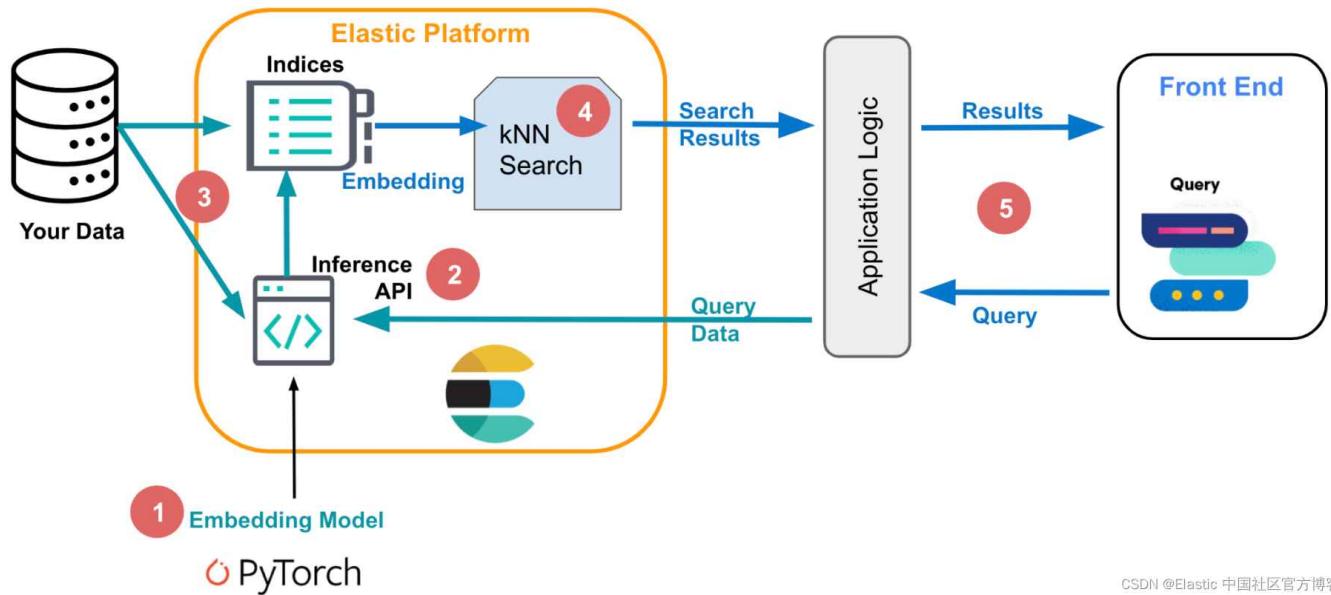
向量DB：查询和索引高维度向量数据

许多嵌入模型可以进行微调以创建良好的语义匹配

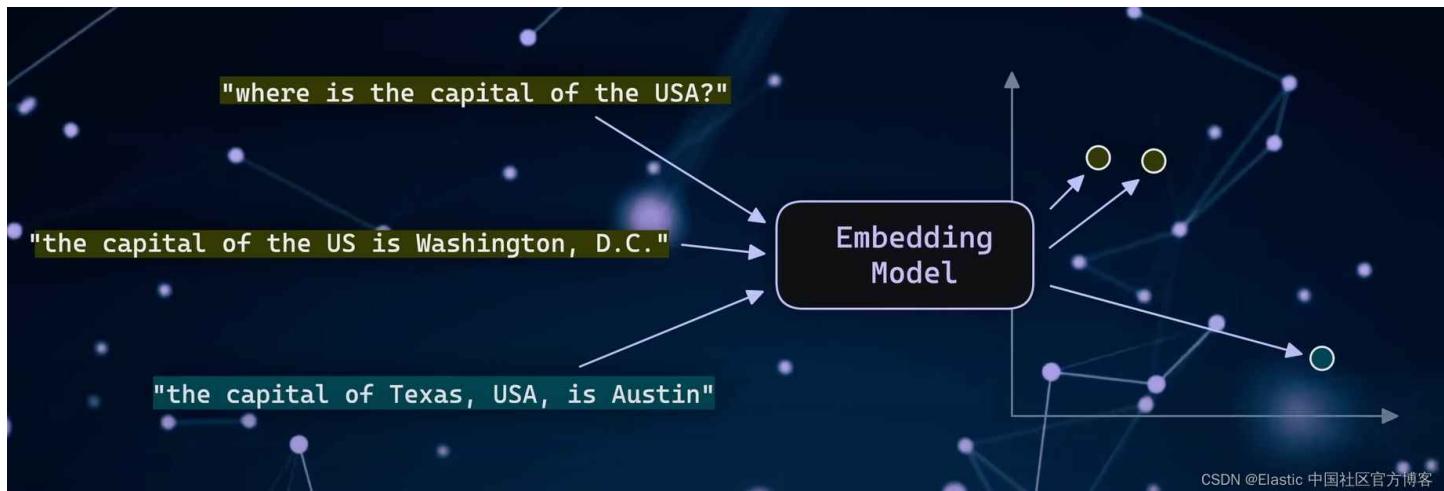
1. 向量相似性搜索 (vector similarity search, VSS) , 或者通常称为语义搜索, 超越了传统的基于关键字的搜索, 允许用户找到可能没有任何共同关键字的语义相似的文档, 从而提供更广泛的结果。向量相似性搜索对密集向量进行操作, 并使用 k-最近邻 (k-nearest neighbour) 搜索来查找相似向量。为此, 首先需要使用文本嵌入模型将文本形式的内容转换为其数字向量表示。所以, 得到嵌入向量后, 就可以进行向量的相似性搜索, 自动筛选出相似度topn 的语料形成上下文, 给生成模型。

## Vector-search 支持的应用程序架构

了解 5 个关键组件



图：向量搜索的流程



图：文本是如何嵌入到向量空间的？

- ElasticSearch: 使用步骤 【Elasticsearch: 如何部署 NLP: 文本嵌入和向量搜索 - CSDN App】  
<http://t.csdnimg.cn/wHq9X>
- Redis: 一个低延迟的向量数据库

向量---加载到Redis---创建索引

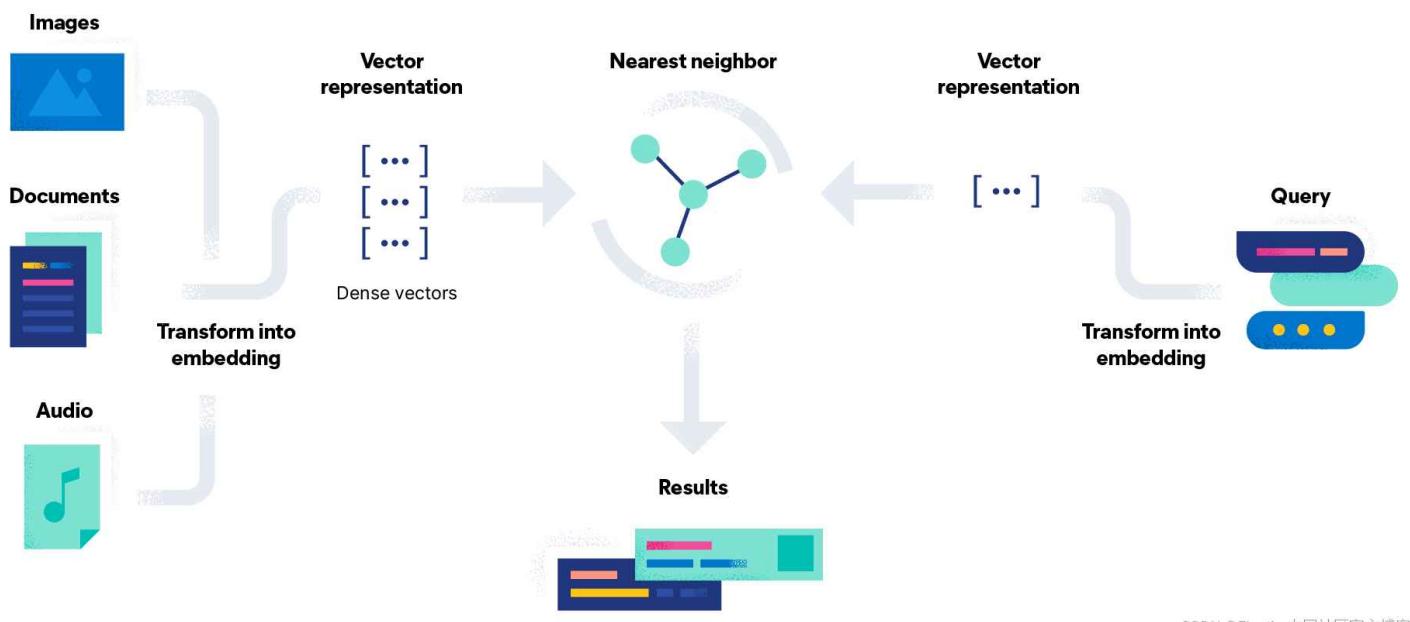
混合查询

传统搜索：限制搜索空间

参考文档：

- VSS Documentation: <https://redis.io/docs/stack/search/reference/vectors/>
- Redis Stack Documentation: [https://partee.io/2022/08/11/vector-embeddings/Redis Stack documentation](https://partee.io/2022/08/11/vector-embeddings/Redis%20Stack%20documentation)

## 嵌入向量--相似度度量



图：文档和用户query的相似度。

## 相似度算法

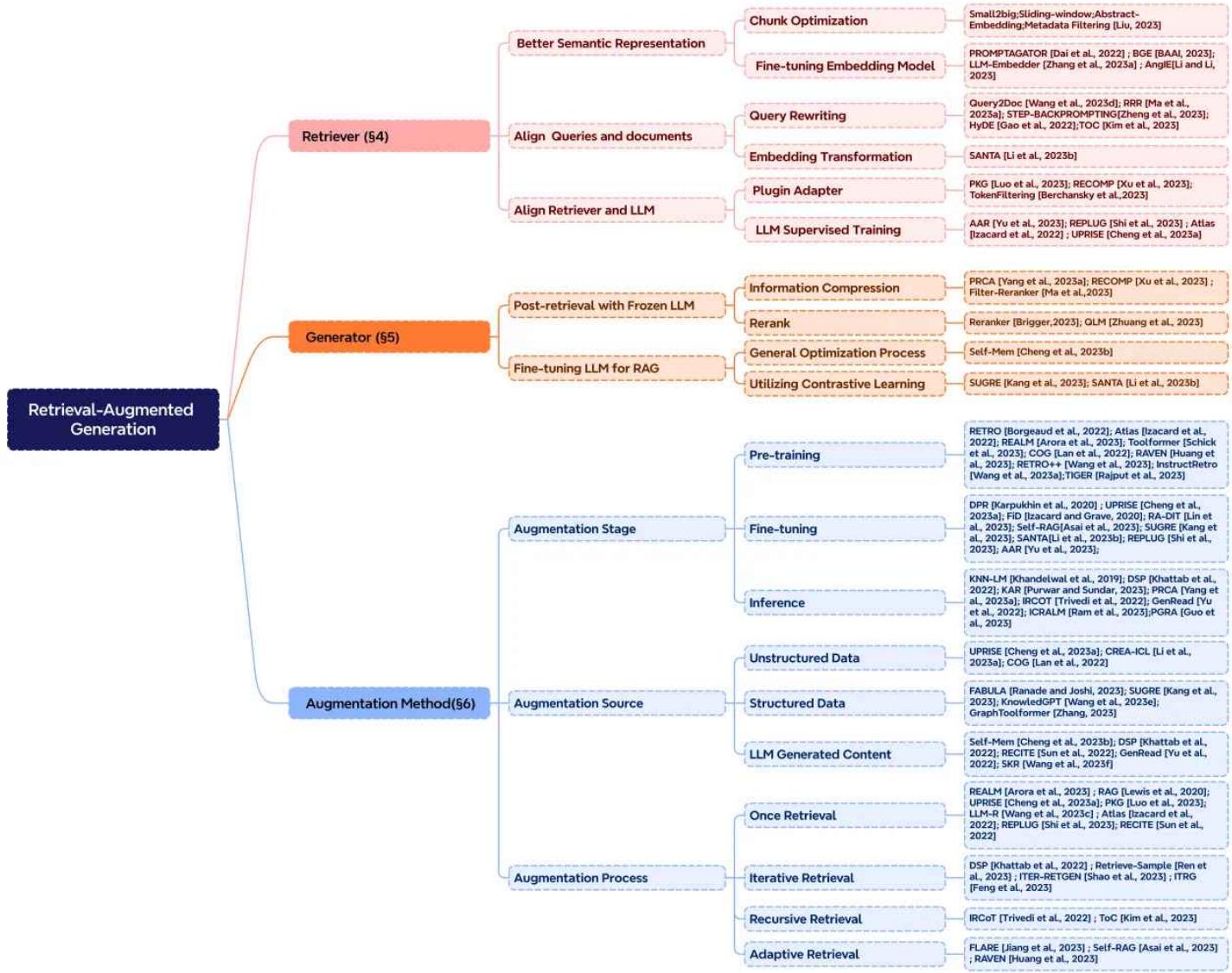
- 欧式距离 L2
- 曼哈顿距离 L1
- 余弦

## 其他检索技术

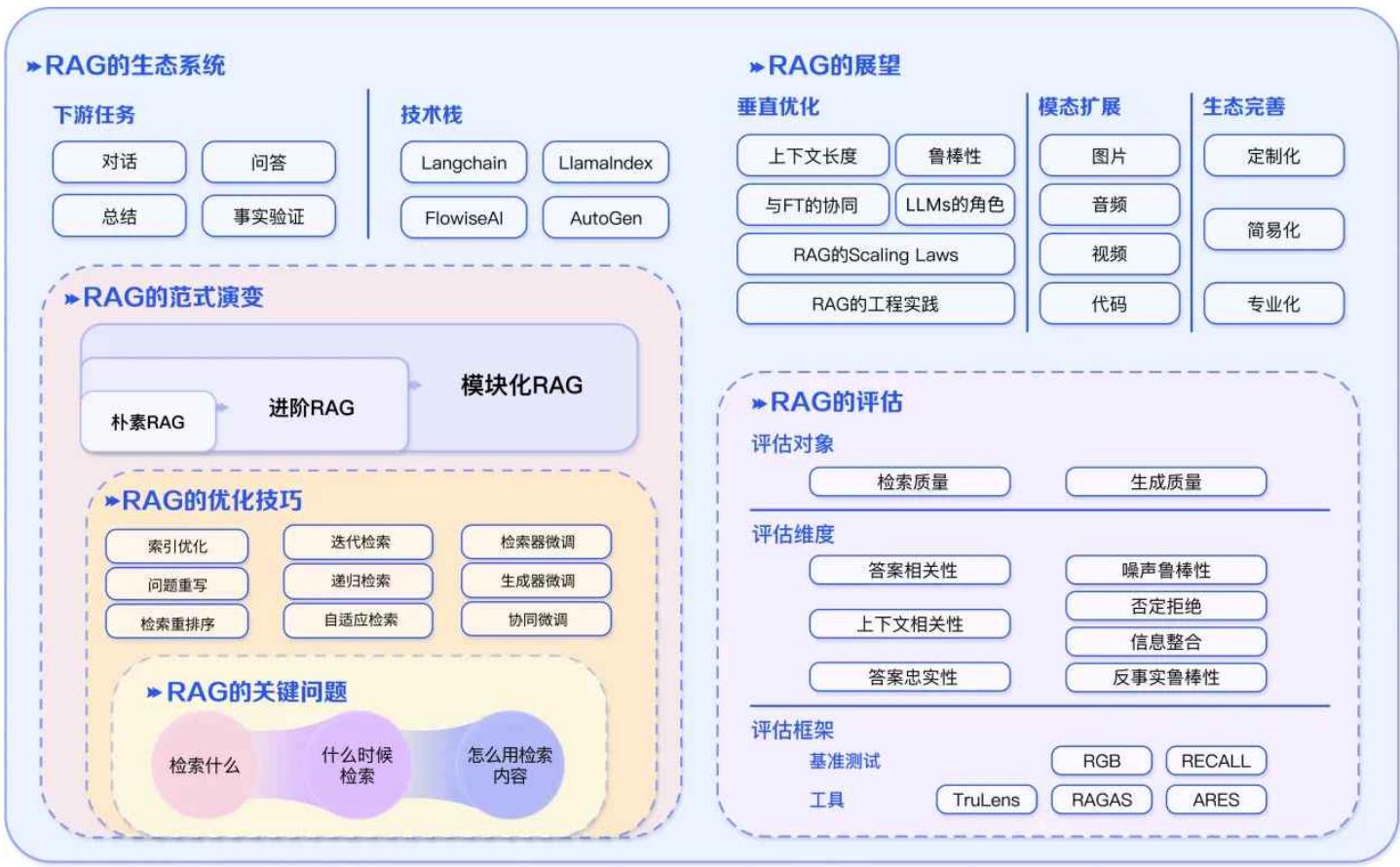
关键词检索 Claude.ai 元数据过滤

SQL检索

<https://github.com/shunliu01/LLM-RAG>



图：RAG技术每个模块的实现路径。具体说，RAG由检索器和生成器组成，围绕这些component可以分别进行优化，包括但不限于对齐文档，文档语义表示增强，使用递归检索技术等，保证生成内容的动态更新、准确性。



图：RAG检索增强技术概览图，分别是RAG的生态系统（包括实现所需技术栈和下游任务）、RAG范式演变、RAG的评估方法，以及RAG技术的展望

## 生成

这一步的重要性相对较低，因为有大量的现有框架可以使用，而且这一步真正发挥作用的是LLM。框架有：Langchain/LlamalIndex/基于Prompt工程的知识库预训练，这部分工作匹配度比较高。

## 工程实现的注意点

分别从研究、教育和生物医学三个领域的case study中，提出在设计RAG系统时需要注意的7个踩坑点。



## 意图识别

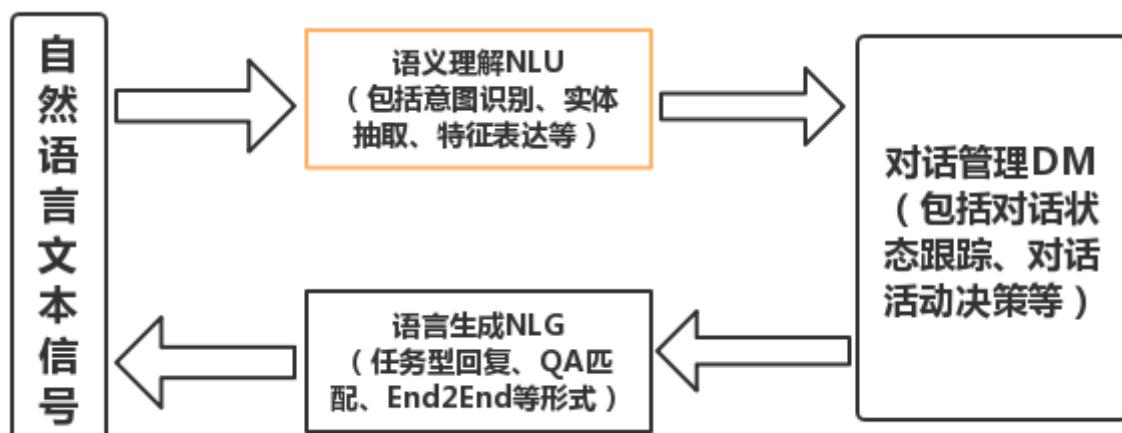
## Intention Recognition

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。



The screenshot shows an Excel spreadsheet titled 'C29知识库语料测试.xls [兼容模式] - Excel'. The table has columns labeled '模块' (Module), '分类' (Category), '分类二' (Category 2), and '结果' (Result). The '问题' (Question) column contains numbered questions. A red box highlights the question '1、发送号码导入数量异常' (1. Number of imported numbers is abnormal) in the '问题' column. The status bar at the bottom indicates '正在说话: 徐雪屹' (Speaking: Xu Xueyi).

图：问题与问题的匹配度、问题与答案的匹配度

## 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)

模块	分类	分类二	结果	问题	备注
1 模块			1	1、上行回复查询和导出功能	
2			1	2、上行回复怎么查询	
3			1	3、客户回复怎么查询	
4			1	4、用户回执的短信怎么查询	
5			1	5、怎么查客户回复	
6			1	6、怎么导出客户/用户回复	
7			1	1、平台发送短信号码导入规则	
8			1	2、在平台上，导入短信号码有哪些规则要遵循？	
9			1	3、平台发送短信号码时，是否有特定的规则限制？如果有，那些规则需要注意？	
10	上行回复		0	1、发送号码导入数量异常	第一次回答不太对，第二次回答对了 (chatgpt) /yyttaominwei-ori/ chatgpt3 python main.py -- --model gpt2 --max-tokens 1000 --temperature 0.7 --repetition-penalty 1.2 --frequency-penalty 1.2 --do_sample True --prompt "Q: 为什么发送号码导入数量异常？A:"
11			1	2、为什么发送号码导入数量异常？	
12			1	3、出现发送号码导入数量异常是什么原因引起的？	
13			1	4、有哪些可能导致发送号码导入数量异常的情况？	
14			1	1、一信通平台黑名单删除	
15			1	2、怎么删除黑名单	
16	黑名单删除		1	3、删除短信黑名单	
17			1	4、平台黑名单删除	
18			1	1、短信删除	
19			1	2、短信短语如何删除	
20	号码导入		1	3、如何进行短信短语删除操作？	
21			1	4、短信短语删除的步骤和方法	
22			1	1、短信库分类标签名称删除	
23			1	2、如何进行短信库分类标签名称删除操作？	
24	短信库		1	3、短信库分类标签名称删除的步骤和方法	
25			1	4、怎么删除短信库分类标签	
26			1	1、为操作员分配扩展接入号功能	
27			1	2、如何为操作员分配扩展接入号？	
28			1	3、扩展接入号怎么分配给操作员	
29			1	4、如何分配扩展号给操作员	
30			1	1、一信通平台增加操作员账号及权限	
31			1	2、如何在一信通平台上增加操作员账号？	
32			1	3、怎样为操作员在一信通平台上创建新的账号？	
33			1	4、在一信通平台上增加操作员账号的步骤和方法？	
34					

图：问题与问题的匹配度、问题与答案的匹配度

## 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。

### FastText

预备知识：

**n-gram**。它是一种基于语言模型的算法，基本思想是将文本内容按照字节顺序进行大小为N的滑动窗口操作，最终形成长度为N的字节片段序列。看下面的例子：

我来到达观数据参观

相应的bigram特征为：我来 来到 到达 达观 观数 数据 据参 参观

相应的trigram特征为：我来到 来到达 到达观 达观数 观数据 数据参 据参观

注意一点：n-gram中的gram根据粒度不同，有不同的含义。它可以是字粒度，也可以是词粒度的。上面所举的例子属于字粒度的n-gram，词粒度的n-gram看下面例子：

我 来 到 达 观 数 据 参 观

相应的bigram特征为：我/来到 来到/达 观数 数据/参 观

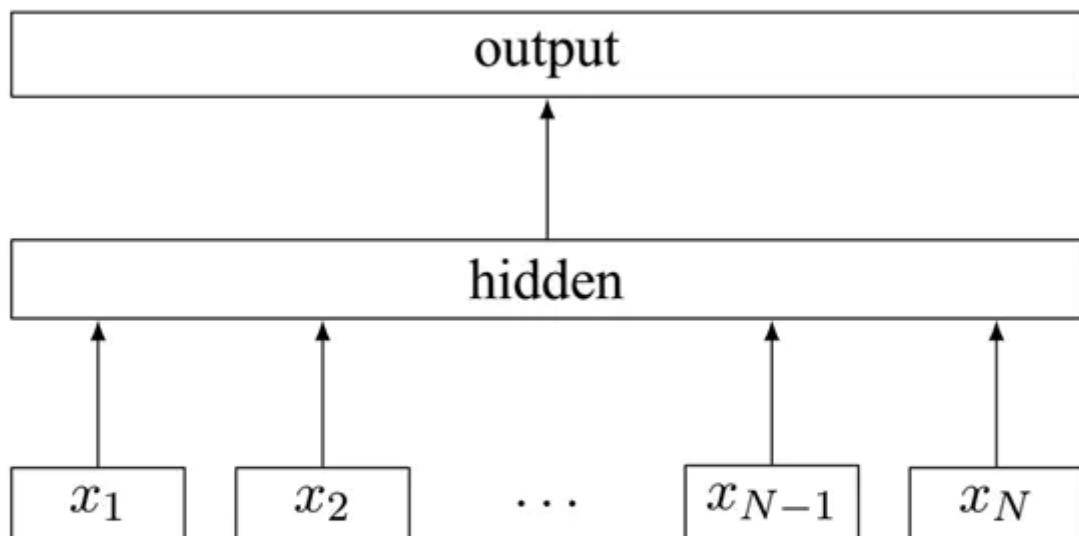
相应的trigram特征为：我/来到/达 观数 来到/达 观数/参 观

n-gram产生的特征只是作为文本特征的候选集，你后面可能会采用信息熵、卡方统计、IDF等文本特征选择方式筛选出比较重要特征。

Hierarchy Softmax Regression(Softmax其实就是多分类的Logistic Regression，相当于把很多个 Logistic Regression组合在一起)

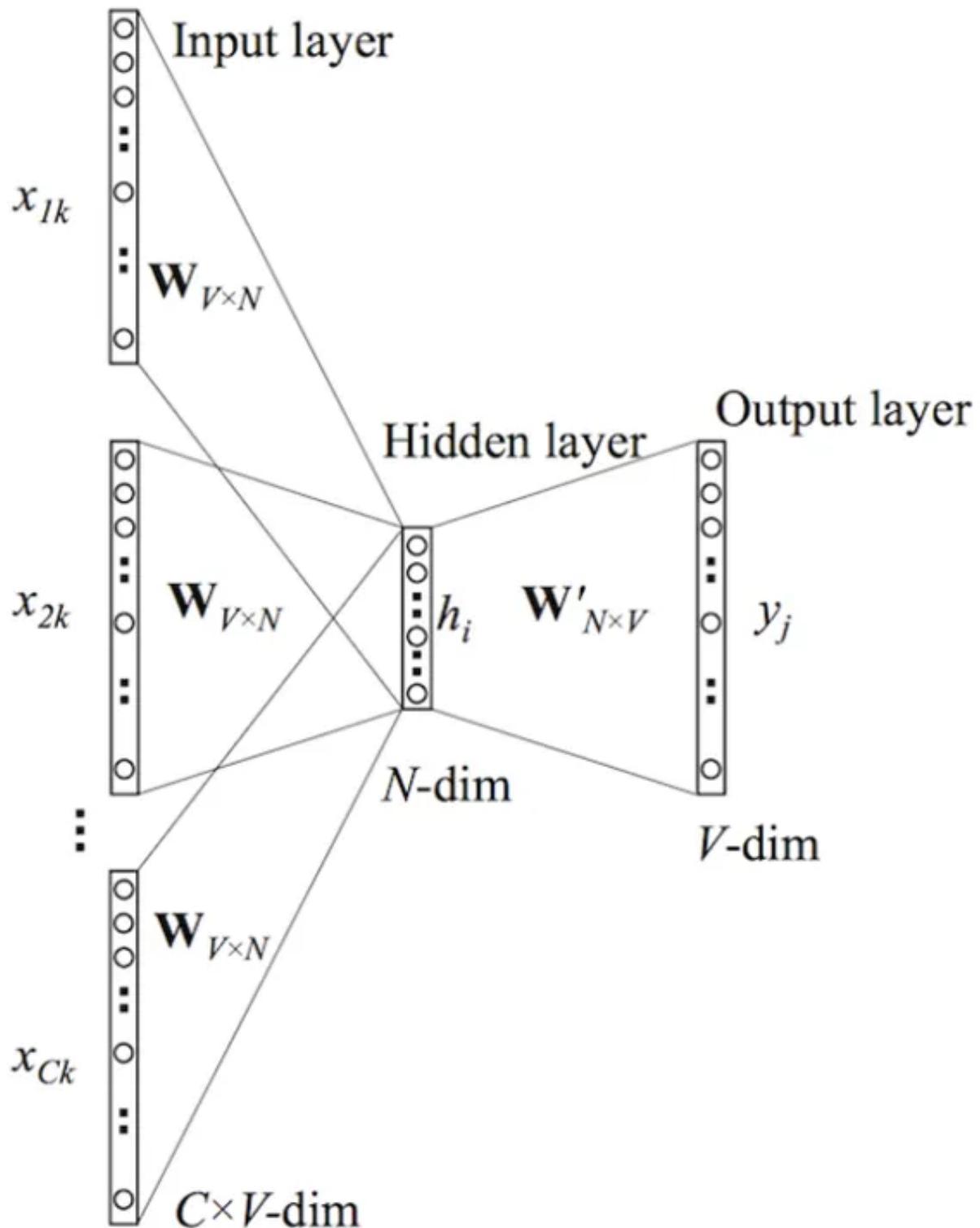
标准的Softmax回归中，要计算 $y=j$ 时的Softmax概率： $P(y = j)$ ，我们需要对所有的K个概率做归一化，这在 $|y|$ 很大时非常耗时。于是，分层Softmax诞生了，它的基本思想是使用树的层级结构替代扁平化的标准Softmax，使得在计算 $P(y = j)$ 时，只需计算一条径上的所有节点的概率值，无需在意其它的节点。

**CBOW**:已知上下文，估算当前词语的语言模型。其学习目标是最大化对数似然函数



**Figure 1:** Model architecture of fastText for a sentence with  $N$  ngram features  $x_1, \dots, x_N$ . The features are embedded and averaged to form the hidden variable.

图：softmax线性多类别分类器，分类器的输入是一个用来表征当前文档的向量；模型的前半部分，即从输入层输入到隐含层输出部分，主要在做一件事情：生成用来表征文档的向量。



图：Word2Vec。1. 训练词向量时，我们使用正常的word2vec方法，而真实的fastText还附加了字符级别的n-gram作为特征输入；2. 我们的输出层使用简单的softmax分类，而真实的fastText使用的是Hierarchical Softmax。

```

# coding: utf-8
from __future__ import unicode_literals

from keras.models import Sequential
from keras.layers import Embedding
from keras.layers import GlobalAveragePooling1D
from keras.layers import Dense

VOCAB_SIZE = 2000
EMBEDDING_DIM = 100
MAX_WORDS = 500
CLASS_NUM = 5

def build_fastText():
    model = Sequential()
    # 通过embedding层，我们将词汇映射成EMBEDDING_DIM维向量
    model.add(Embedding(VOCAB_SIZE, EMBEDDING_DIM, input_length=MAX_WORDS))
    # 通过GlobalAveragePooling1D，我们平均了文档中所有词的embedding
    model.add(GlobalAveragePooling1D())
    # 通过输出层Softmax分类(真实的fastText这里是分层Softmax)，得到类别概率分布
    model.add(Dense(CLASS_NUM, activation='softmax'))
    # 定义损失函数、优化器、分类度量指标
    model.compile(loss='categorical_crossentropy', optimizer='SGD', metrics=['accuracy'])
    return model

if __name__ == '__main__':
    model = build_fastText()
    print(model.summary())

```

## 2. 意图识别结果

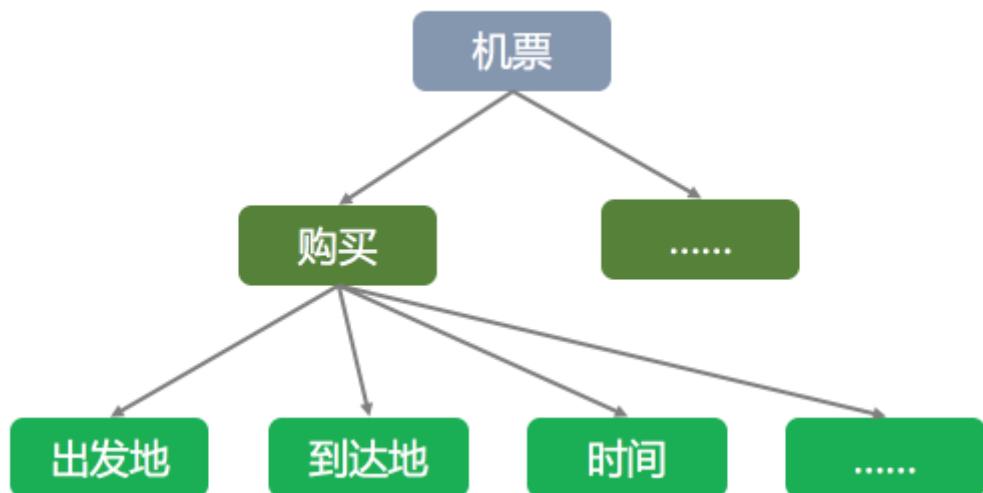
意图识别之后，我们能够得到3类问题，根据类别采用不同的决策，如果是任务型的，可以分配到具体的任务意图处理。

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



QA系统中，意图识别的地位等同于对用户的问题进行划分，并生成不同形式的回答

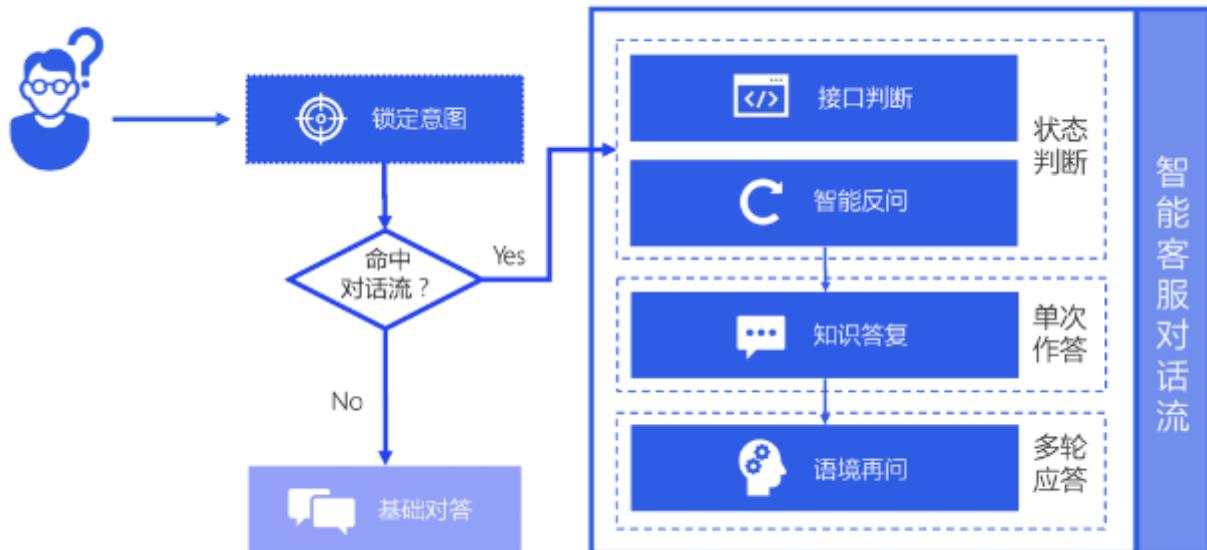
b) 任务型：例如“我想订一张明天从杭州到北京的机票”→ 意图决策+槽位填充的匹配方式。槽位填充通常需要使用对话管理技术，如对话状态跟踪、对话策略生成等。



任务型对话中，模型关注event schema(who/where/when/why)。比如，我想订一张北京-上海机票，模型思考的是如何确定旅途的src/dest/time等，提取用户输入中与意图相关的实体信息，确定用户的意图

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



基于分类维护一个意图堆，确定意图边界和具体的意图切换和修改逻辑，互斥，相似，上下位在多轮交互中用户会不断的添加或修改意图的子意图，需要维护一份当前识别的意图集合。

- 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

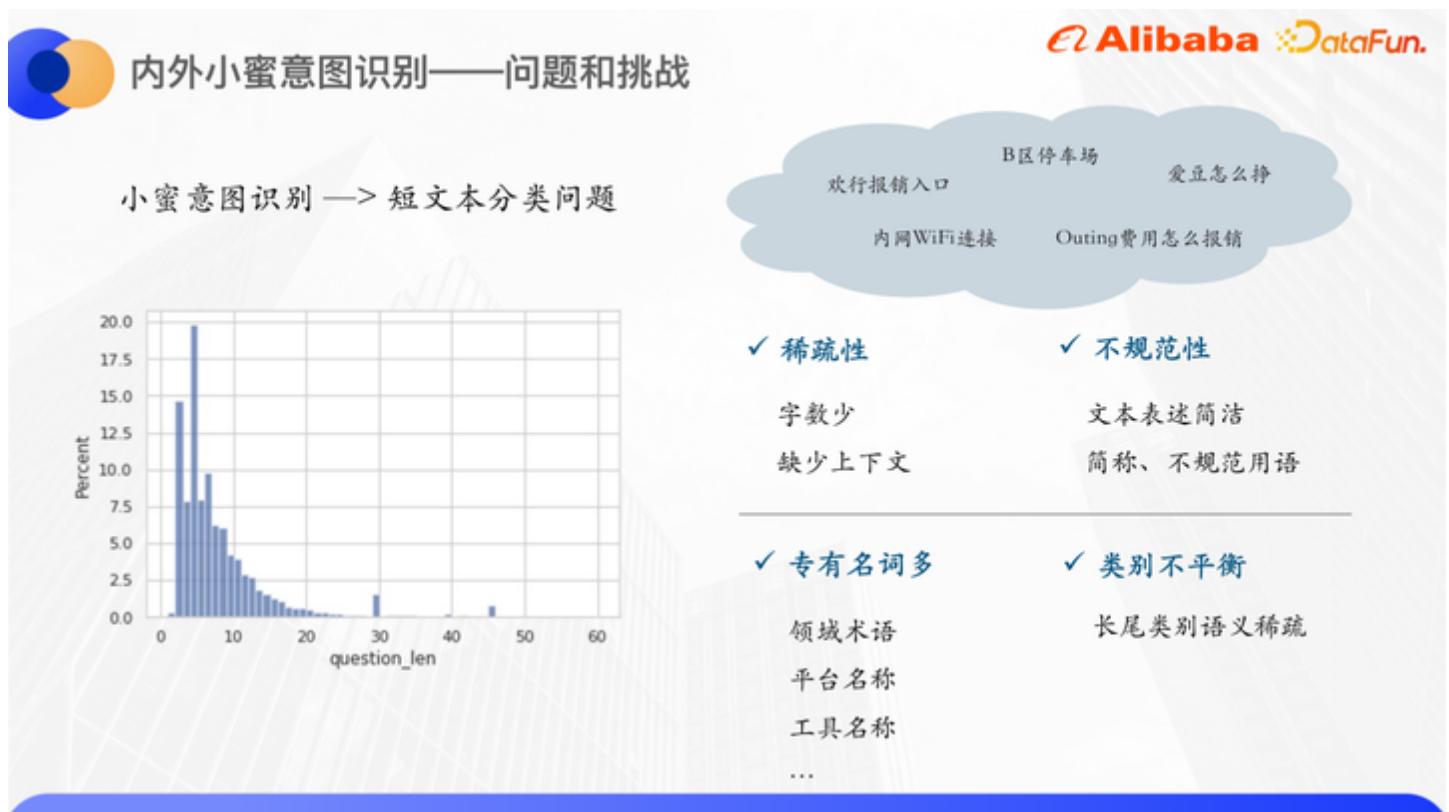
## 4. 开源的意图识别模型

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



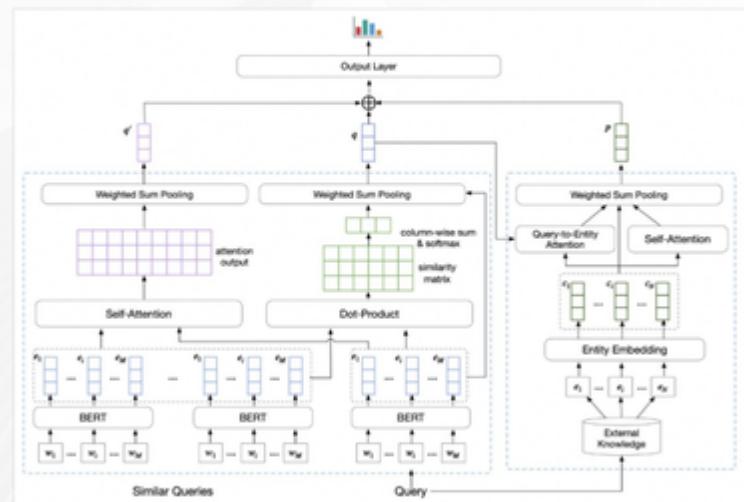
图：用向量召回的方式，召回知识卡片的实体和相似 query。用 Bert 分别计算 query 和知识卡片文本描述的具体量。一般来说不会直接使用 Bert 的 CLS 向量作为句子表征，很多论文中也有提到，直接使用 CLS 向量作为句子表征效果会很差，因为 Bert 输出的向量会出现表达退化的问题，不适合直接用它做无监督的相似度计算，所以使用对比学习的思想，把相似样本拉近，让不相似的样本尽量均匀分布。

具体来讲，就是在数据集上 finetune 了一个 Sentence-Bert，其模型结构和训练方式可以产出比较好的句向量表征。它是一个双塔的结构，两边的 Bert 模型共享模型参数，两个句子分别输入到 Bert 里面，Bert 输出的 hidden states 做 pooling 之后会得到两个句子的句向量。这里优化的目标是对比学习的 Loss，infoNCE。



## 内外小蜜意图识别——文本分类

Alibaba DataFun.



### Entity信息融合

$$a_i = \text{softmax}(\gamma\alpha_i + (1 - \gamma)\beta_i)$$

$$p = \sum_{i=1}^N a_i c_i$$

### Query信息提取

$$\theta = \text{softmax}(\text{sum\_pooling}(M))$$

$$q = \sum_{l=1}^M \theta_l E_l^q$$

### Query-Similar & Query信息融合

$$E^c = \text{concat}([E^q, E^s])$$

$$Q' = \text{Attention}(E^c, E^c, E^c)$$

$$q' = \text{sum\_pooling}(Q')$$

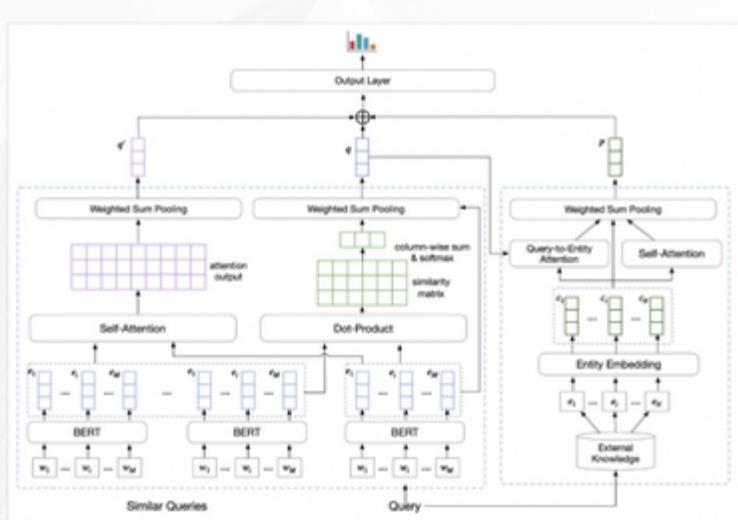
图：上图是文本分类的模型结构，在 Encoding 层使用 Bert 提取原始 query 和相似 query 的词向量的表征，每个知识卡片的实体会维护一个实体的 ID embedding，ID embedding 是随机初始化的。模型结构图右侧，用来处理 query 召回的实体，得到实体的统一向量表征。因为短文本本身比较模糊，所以召回的知识卡片实体也会有一定的噪声，通过使用了两个 attention 的机制，让模型更加注意正确的实体。

一个是 Query-to-Entity 的Attention，目的是让模型更注意和 query 相关的实体。另一个是实体本身的 self Attention，可以把互相之间相似的实体权重提高，降低噪声实体的权重。综合两组Attention权重，就得到最终实体的一个向量表示。



## 内外小蜜意图识别

Alibaba DataFun.



### 离线指标

Model	Accuracy
RCNN	86.7%
TextCNN	87.81%
BiLSTM-SA	87.87%
STCKA	88.23%
BERT finetune	90.86%
SQKA	88.47%
<b>SQKA-BERT</b>	<b>91.49%</b>

### 在线指标

➤ 内外小蜜2.0上线后，精准命中率从63.94%（2022年10月）提升到**75.79%**（2022年3月），用户反馈中“无帮助”占比相对下降**12%**。

实验结果，超过了 BERT finetune 的结果，编码层不用 Bert 的话，也是超过了所有非 Bert 的模型。

**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



延陵季扎 LvS

3小时前

在特定场景下，小样本（1W）、类别极不平衡的短文本多分类，bert的泛化效果没简单模型好，导师怎么看？



收起



4



⋮

[查看视频详情](#)



Itf腾飞，只会写C语言 等3个新点赞



只会写C语言 LvS

2小时前

极端一点，如果只有两个训练样本，什么模型也不如一条直线。



3



⋮



跟李沐学AI LvS UP

1小时前

bert论文模型是在长文本训练，对短文本效果一般很正常。需要找合适的pretrain模型



1



⋮

知乎 @延陵季扎

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

ML

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“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。类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。



## 企业搜索场景

Alibaba DataFun.

### 企业统一搜索

The screenshot shows the internal search interface for Alibaba. At the top, there's a search bar with the placeholder '请输入关键词' and a magnifying glass icon. Below it, a banner for '阿里巴巴春招权益指南' is displayed. The main search results area shows a list of items, with one item highlighted in blue. The bottom part of the interface includes a navigation bar with links like '综合', '联系人', '群组', '聊天记录', '文档', '日志', '标签', and '功能'.

### 企业员工助手

The screenshot shows the 'Enterprise Employee Assistant' interface. It features a search bar at the top with the placeholder '请输入关键词'. Below the search bar, a search result for '停车场' (Parking Lot) is shown, including a small image of a parking lot and some descriptive text. The interface has a clean, modern design with a white background and blue accents.

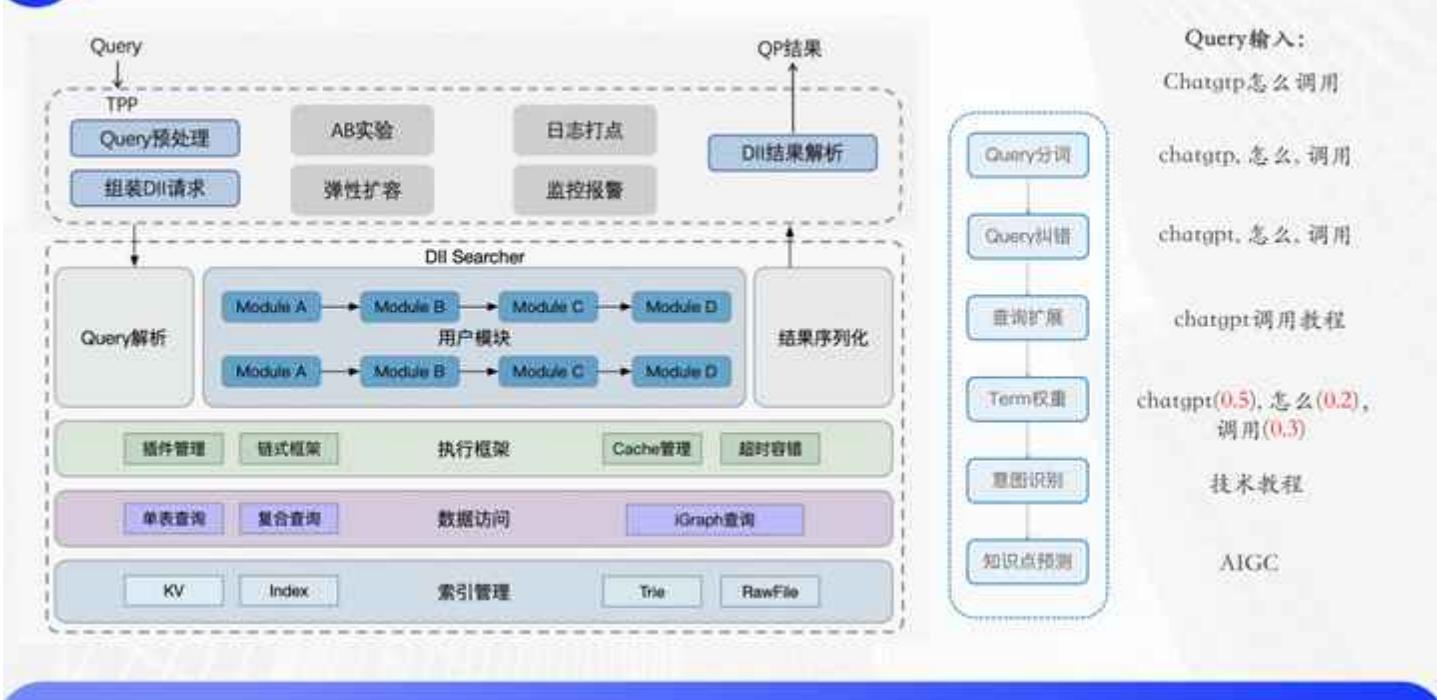
### 行业搜索

The screenshot shows the 'Industry Search' interface for Alibaba. It includes a search bar with the placeholder '请输入关键词' and a search button. Below the search bar, there are several categories or filters: '我要采购' (I Want to Buy), '实物采购' (Physical Procurement) with sub-options like '商品' (Products), '耗材' (Consumables), '办公用品' (Office Supplies), and '生产设备' (Production Equipment); and '生产运营' (Production Operation) with sub-options like '工厂管理' (Factory Management), '车间管理' (Workshop Management), and '质量控制' (Quality Control). The overall layout is organized and professional.

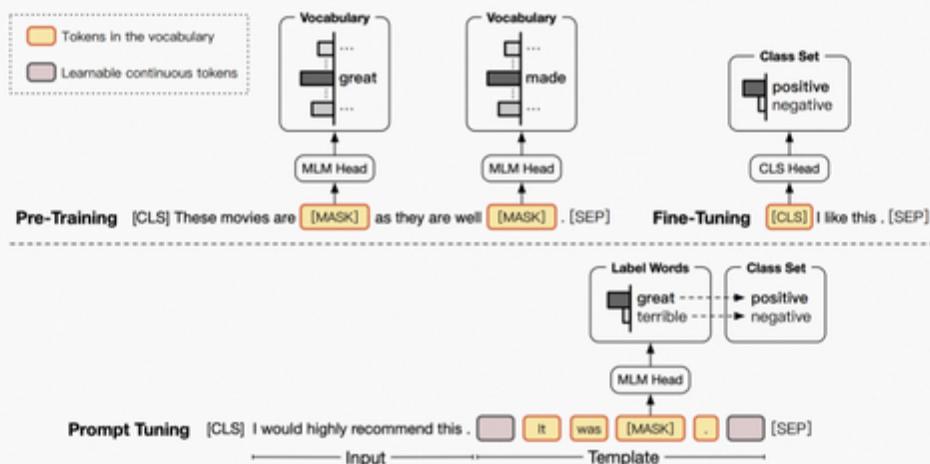
## 企业搜索架构

Alibaba DataFun.





## Prompt Learning



- **预训练任务:** Masked Language Model
- **微调:** 分类任务
  - 未使用label本身的语义信息
  - 和预训练任务不一致
  - 需要较多的标注数据

- **Prompt Learning**
  - 分类任务 → MLM任务
  - 适合zero-shot 和 few-shot
  - 支持label扩展

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。

## 挑战



## 企业场景Query理解的挑战

- 领域知识不足
- 有标注数据少

## 知识增强的多种方式

- 预训练模型
- 知识图谱
- 数据库

## 企业场景下基于prompt的文本分类

- 现有预训练模型zero-shot的局限性
- template设计

## 企业预训练大模型

- 预训练数据来源
- 所有任务 → 语言模型任务
- 准确性如何保证

# Rasa: 大语言模型用于意图识别

任务：

few-shot的意图识别

open vocabulary的意图识别

挑战

- 数据稀疏性
- 构建领域QA系统，需要提供领域知识
- 用户query通常是短文本，上下文信息不足
- 用户意图数据标注成本高，多为few-shot甚至zero-shot场景

参考：

[基于知识增强和预训练大模型的 Query 意图识别 - 知乎 \(zhihu.com\)](#)

[大模型+知识库/数据库问答实践过程的经验汇总（三） - 知乎 \(zhihu.com\)](#)

[详解小样本短文本多分类-对比TextCNN与ALBERT分类效果的实践:  
https://www.zhihu.com/tardis/zm/art/443782891?source\\_id=1005](#)

由浅到深的文本分类综述：<https://zhuanlan.zhihu.com/p/596112080>

5NLP智能客服：

<https://github.com/polygraph8/5minNLP/blob/master/5%E5%88%86%E9%92%9FNLP-%E6%99%BA%E8%83%BD%E5%AE%A2%E6%9C%8D.md>

多轮对话机器之话题意图识别<https://cloud.tencent.com/developer/article/1462181>



知乎 <https://zhuanlan.zhihu.com/p/630723604>

## 基于知识增强和预训练大模型的 Query 意图识别

导读：Query 意图识别，是 Query 理解里面一个很重要的任务，对下游搜索的相关性、准确性有很大的影响。本文将分享基于知识增强和预训练模型的 Query 意识…