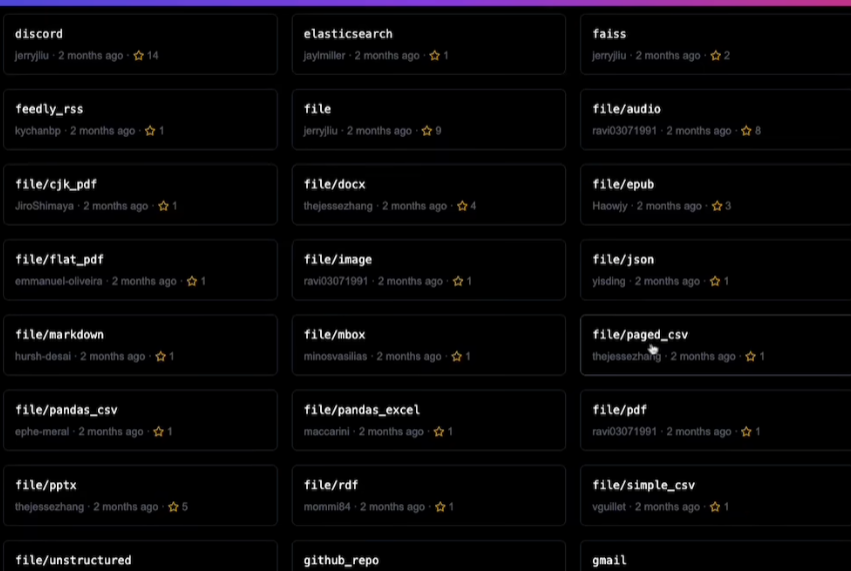
如果我们要做Context\_learning的话，就要解决数据获取的问题

Llama\_index提供了很多连接各种数据平台的connector



通过connector拿到数据之后，就需要进行数据分割

分割之后就需要一个地方去存这些数据，这就涉及到一个的概念index，分割好数据之后形成一个chunk，就数据存到一个地方，当我需要查询的时候，我就从一个数据结构里面查出来，那这个数据结构就是index，index就只有两个，一个存储数据，一个是查询功能，查询功能意思就是实现了一些查询接口，这个index就有几种类型，我们常用的index就是基于vector，我们可以将数据转成各种形式例如vector类型，矩阵类型（图像识别）。

Llama将数据load进来之后，数据是放在document对象里面的。Document是完整的数据集。Document被分割之后就形成了chunk的这种node结构，其实node就是一个个chunk，它里面就是被分割后的一段段文本，存到vector store里面，就是一个个node，然后就将一个个node节点embedding之后变成了一个向量，一个node对应一个文本的向量。Vector他也是支持很多第三方存储技术的，我们可以直接存到内存里面去，但是内存一般是比较暴力的一种方式，他会去比较每一个向量的距离就是余弦值。当计算量非常大的话，我们是需要第三方技术去实现这个向量的查找。开源的face这个向量数据库。

Llama\_index有很多的index，对文章对不同操作的时候就需要不同的index，对文章作总结的index，找出最匹配问题的段落的index，

[Summary Index](https://gpt-index.readthedocs.io/en/stable/api_reference/indices/list.html)

[Table Index](https://gpt-index.readthedocs.io/en/stable/api_reference/indices/table.html)

[Tree Index](https://gpt-index.readthedocs.io/en/stable/api_reference/indices/tree.html)

[Vector Store Index](https://gpt-index.readthedocs.io/en/stable/api_reference/indices/vector_store.html)

[Structured Store Index](https://gpt-index.readthedocs.io/en/stable/api_reference/indices/struct_store.html)

[Knowledge Graph Index](https://gpt-index.readthedocs.io/en/stable/api_reference/indices/kg.html)

[Empty Index](https://gpt-index.readthedocs.io/en/stable/api_reference/indices/empty.html)

Langchain的最重要的agent，我们以前想让gpt对接某些操作系统，我们要指定gpt要做哪一步哪一步，但是还有一种方式就是我给gpt很多能力，让它自己去决定调用哪些方法。

Langchain可以从很多结构化数据或者非结构化中去读取数据，

Langchain里面的chain包含两个内容，一个是模型，一个是prompttemplate，可以将多个语言模型融合起来，相当于gpt做分类功能，吐出prompt之后之后传递给更加专业的llm。

盲猜什么是langchain的agent，就是为了实现不同目的，就是设计一系列的prompt，然后利用gpt的逻辑推理能力，最终达到用户的目的。

Langchain里面的核心其实也是和gpt交互多次，写好固定的输出模板，然后调用规定的tools获取数据，然后完成任务。