Long context4 提示压缩1

背景知识

参考文献：Empirical distribution

信息量：

设随机变量X的概率密度函数为p(X)，对于 $X = x$ 这件事的信息量为：

$$I(x) = -log_2 p(x)$$

log底数为2表示用二进制数据对x进行编码时，所需要的bit数。越罕见的事件，传达的信息越多。

信息熵

随机变量X的平均信息量，定义为：

$$H(x) = -\sum_{x \in \mathcal{X}} p(x) log_2 p(x)$$

其中 $\mathcal{X}$ 时X的样本空间。X的分布越均匀，事件的不确定性越大，需要的存储空间越大。

信息熵也可以定义为：根据真实分布，我们能够找到一个最优策略，以最小的代价消除系统的不确定性（比如编码），而这个代价的大小就是信息熵

交叉熵和困惑度

设两个定义在相同样本空间上的概率分布 p 和 q，假设p为真实概率分布，q是对p的近似。使用q来衡量随机变量X的信息熵为：

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) log_2 q(x)$$

即p和q之间的交叉熵，交叉熵定义为：衡量在给定的真实分布下，使用非真实分布指定的策略消除系统的不确定性所需要付出努力的大小。

KL散度（相对熵）是用来衡量两个概率分布之间的差异：

$$D_{KL}(P||Q) = \sum_i P(i) \log_a \frac{P(i)}{Q(i)}$$

定义概率分布的perplexity为：

$$PPL(q) = 2^{H(p,q)}$$

现实中，当真实概率分布未知时，采用经验分布替代真实概率分布。在样本集 $\{x_i\}_{i=1}^n$ 上，经验分布定义为：

$$\hat{F}_n(x) = \begin{cases} 0, & if \quad x < x_1 \\ \dfrac{i}{n}, & if \quad x_i \le x < x_{i+1} \\ 1, & if \quad x \ge x_n \end{cases}$$

其中 $\hat{F}_n(x)$ 为累积概率分布函数，根据 **Glivenko–Cantelli 定理**，当 $n \to \infty$ 时，经验分布函数收敛到真实分布函数。基于经验分布函数有 $P(X = x) = \dfrac{1}{n}$

$$PPL(q) = 2^{-\frac{1}{n}\sum_{x\in\mathcal{X}} log_2 q(x)}$$

$$PPL(q) = 2^{-\frac{1}{n} log_2 \prod_{x\in\mathcal{X}} q(x)}$$

$$PPL(q) = \left(\prod_{x\in\mathcal{X}} q(x)\right)^{-\frac{1}{n}}$$

语言模型的困惑度

语言模型可以看作是给定一个token序列，根据已知的token来预测下一个token出现的概率。整个句子出现的概率为：

$$p(X_1, X_2, ..., X_m) = p(X_1)p(X_2|X_1)...p(X_m|X_1, ...X_{m-1})$$

对于长度为m的序列，其困惑度为：

$$PPL = \left(\prod_{i=1}^{m} q(X_i|X_1, ..., X_{i-1})\right)^{-\frac{1}{n}}$$

PPL越小说明q越大，即我们期望的序列出现的概率就越高。一般模型越大、训练数据越多越丰富，复杂度越低。

N-gram

定义：第n个词出现与前n-1个词相关，而与其他任何词不相关。整个句子出现的概率就等于各个词出现的概率乘积。各个词的概率可以通过语料中统计计算得到。n越大perplexity越低。

N-gram模型也是一种语言模型，假设句子S是有词序列w1,w2,w3...wn组成，则采用n-gram时的出现概率为：

$$P(S) = P(w1, w2, w3, ..., wn) = P(w1)P(w2|w1)P(w3|w1, w2)...P(w_n|w_1, w_2, w_3..., w_{n-1})$$

存在问题：1. 参数空间过大（时间和空间复杂度都是词表的指数次方级别）。 2. 数据稀疏严重（某些词同时出现的情况可能没有）。

为了解决第一个问题，引入马尔科夫假设（Markov Assumption）：一个词的出现仅与它之前的若干个词有关。常用的往往是1-gram（unigram），2-gram（bigram）和3-gram（trigram）。

条件概率的计算采用最大似然估计， 根据大数定理，只要统计量足够，相对频度就等于概率。 通常用相对频率作为概率的估计值。这种估计概率值的方法称为最大似然估计。

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

为了解决第二个问题，有两种方法：

- 数据平滑：使所有的N-gram概率之和为1，使所有的n-gram概率都不为0。通过重新分配整个概率空间，使已经出现过的n-gram的概率降低，补充给未曾出现过的n-gram，使得概率分布尽量均匀。（对于没有看见的事件，我们不能认为它发生的概率就是零，而需要分配给一个较小的概率）。
- 回退插值：当我们无法计算高阶的n-gram时，可以回退到低阶的n-gram。比如对于3-gram，我们可以用2-gram和1-gram进行插值估算。

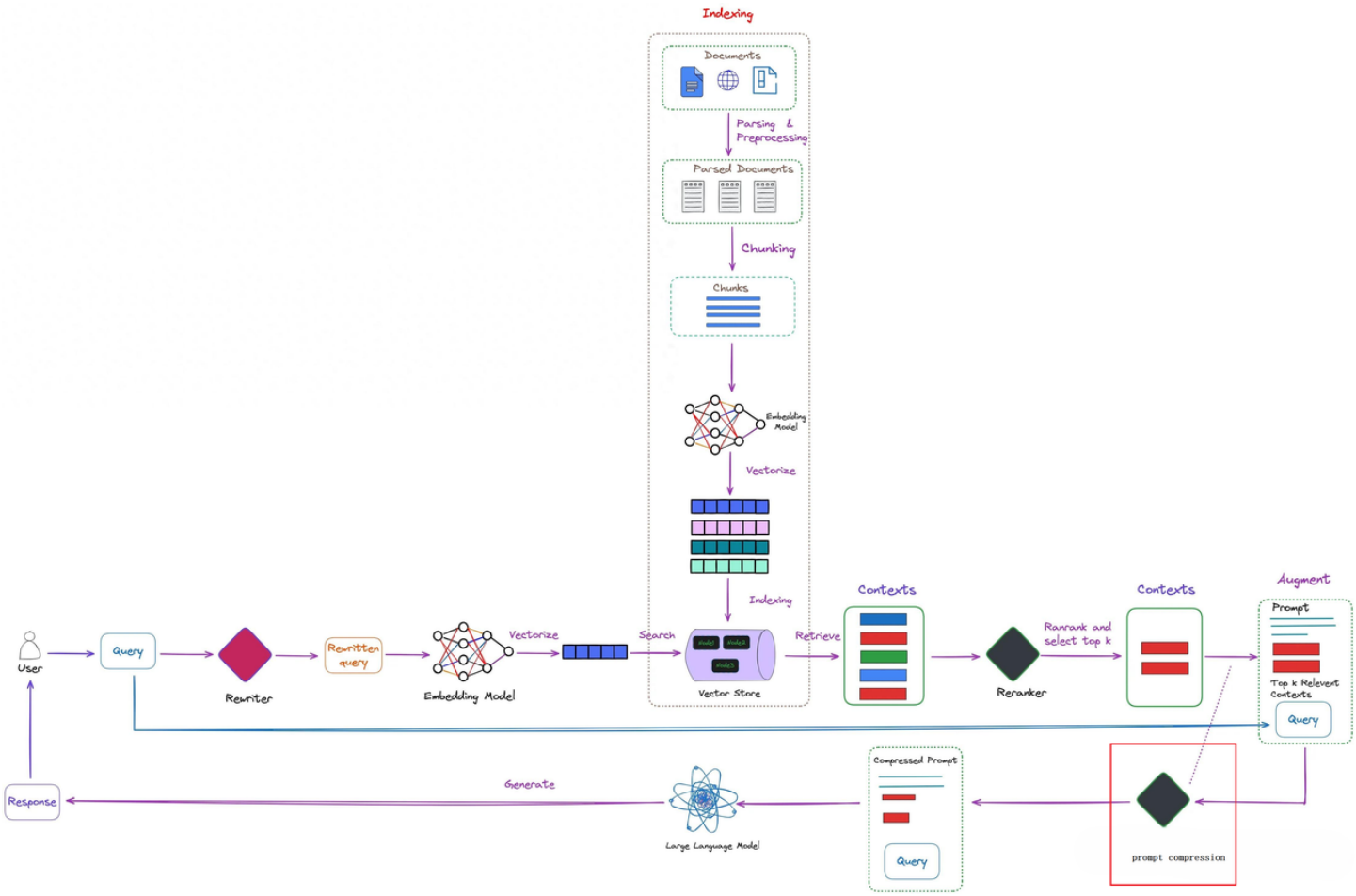
$$P(t_i|t_{i-1}t_{i-2}) = \lambda_3\hat{P}(t_i|t_{i-1}t_{i-2}) + \lambda_2\hat{P}(t_i|t_{i-1}) + \lambda_1\hat{P}(t_i)$$

$$P(t_i|t_{i-1}) = \lambda\hat{P}(t_i|t_{i-1}) + (1-\lambda)\hat{P}(t_i)$$

$$P(t_i) = \lambda\hat{P}(t_i) + (1-\lambda)\frac{1}{N}$$

对于LLM，越详细的prompt，往往效果越好。但是当prompt(在下图的Rag场景中，上下文一般指prompt)长度过长时，一方面耗时更长；另一方面由于检索到的很多内容可能与答案无关，LLM需要结合context对rerank重新排名后的context进行理解并生成答案，当context长度过长时效果较差。

如下图所示，右下角的prompt提示压缩能有效解决这些问题，只保留prompt中有价值的token。



prompt压缩主要可以分为以下几类：

- LongLLMLingua：基于信息熵，使用小模型计算原始prompt中每个token的困惑度，删除困惑度较低的标记。
- 基于软提示调优，在下游领域对LLM进行微调，但不能应用于黑盒LLM。
- 对LLM进行数据蒸馏，训练模型生成可解释性强的摘要，适用于黑盒LLM。

Selective Context

信息量：量化事件传达的信息量，设随机变量X的概率密度函数为p(X)，对于 $X = x$ 这件事的信息量为：

$$I(x) = -log_2 p(x)$$

越罕见的事件，传达的信息越多。可以通过小的语言模型计算prompt中每个token的信息量，接着将token合并成句子/短语，合并后句子/短语的信息量为每个token的和。之后将句子/短语按照信息量降序分布，只保留包含前p%信息量的句子/短语。

Selective Context忽略了压缩后context内容之间的连接性，并且没有考虑LLM与压缩prompt的小模型的相关性。