技术博客：知识点整理与总结

徐善若

2025 年 5 月 21 日

摘要

本文基于与用户的对话内容整理出多个技术领域的知识点，涵盖 ** 异步编程、数据库设计、Transformer 架构、提示词设计、SDK、IDE、感受野与窗口切分等 **，旨在为读者提供一个全面且有条理的技术博客，特别是为日后自己和他人做参考。每个知识点将辅以实例和公式，以便更加生动易懂。

目录

1 Transformer 架构与 LSTM 的对比

1.1 Transformer 架构的基本原理

Transformer 是一种基于 *自注意力机制*（*Self-Attention*）的深度学习架构，通常用于自然语言处理（NLP）任务。与传统的 *LSTM* 或 *RNN* 不同，Transformer 能够并行处理输入序列中的所有元素，且具有全局感受野。

1.1.1 关键特性

- **自注意力机制**：每个词在处理时可以与序列中的其他词进行交互，捕捉长距离依赖关系。
- **并行处理**：在训练阶段，Transformer 可以对输入的所有词并行处理，这比 LSTM 中按顺序逐步处理的方式要高效得多。
- **全局感受野**：通过自注意力机制，模型能够在每个时间步对整个输入序列进行感知，不再依赖局部的感受野。

1.1.2 示例公式

在自注意力机制中，对于输入序列 $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$，每个位置的注意力计算可以用以下公式表示：

$$\text{Attention}(\mathbf{X}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

其中：- $\mathbf{Q}$ 是查询矩阵（Query）- $\mathbf{K}$ 是键矩阵（Key）- $\mathbf{V}$ 是值矩阵（Value）- d_k 是键的维度。

1.2 LSTM 的工作原理

LSTM（长短时记忆网络）是一种特殊的 *RNN*（循环神经网络），能够有效解决传统 RNN 中的梯度消失和梯度爆炸问题。LSTM 在每个时间步通过引入记忆单元来处理时间序列数据。

1.2.1 关键特性

- **顺序处理**：每次只能感知当前时间步及其之前的状态，感受野是逐步扩展的。
- **局部感受野**：LSTM 只能处理当前时间步及之前的状态，对于长序列，它的感受野相对较小。

1.2.2 示例公式

LSTM 的核心是引入了记忆单元和多个门控机制，下面是 LSTM 中的输入门、遗忘门、输出门的计算公式：

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$h_t = o_t \cdot \tanh(c_t)$$

1.3 Transformer 和 LSTM 的对比

特性	Transformer	LSTM
处理方式	并行处理，使用自注意力机制	按顺序逐步处理，每次使用前一个时间步的输出
感受野	全局感受野，能够感知整个输入序列	局部感受野，需要逐步扩展
计算效率	高效并行计算，适合长序列处理	计算较慢，依赖顺序计算
适用场景	长文本、序列标注任务、机器翻译等	序列生成、语音识别等

1.4 GPT 的生成方式

GPT（Generative Pre-trained Transformer）是一个自回归的语言模型。尽管它在训练时采用并行处理，但生成文本时是按顺序生成的。每次生成一个词时，模型会依赖之前生成的词作为上下文。

—

2 提示词（Prompt）设计

2.1 提示词的构造原理

在与大语言模型（LLM）交互时，提示词的构造至关重要。提示词（Prompt）是用来引导模型生成特定内容的输入，结构化的提示可以帮助模型生成更准确、更符合要求的输出。

2.1.1 提示词的基本结构

- **背景信息**：提供任务背景和上下文，帮助模型理解任务目标。
- **具体指令**：明确告诉模型要执行的具体操作或生成内容的格式要求。

2.1.2 示例：

背景信息：你是一个环保领域的专家，专注于全球变暖问题。具体指令：请列举全球变暖的主要原因，并简要说明其影响。

2.2 上下段拆分的意义

根据模型的处理方式，提示词可以拆分为 ** 上半段（背景信息）** 和 ** 下半段（任务指令）**，这样有助于清晰地设定任务框架，确保模型理解任务背景后再执行具体指令。

- 先背景信息，再具体指令：通常，这样的顺序更清晰，有助于模型理解上下文并生成有针对性的内容。
- 先指令，再背景信息：在某些任务明确、背景简单时，顺序可以颠倒，优先给出任务指令。

2.2.1 例子：

背景信息：你是一名历史学家，专注于中国古代历史。任务指令：请简要说明秦朝的建立及其历史影响。

—

3 响应体（Response Body）与 LLM 的关系

3.1 响应体的定义

响应体（Response Body）是指服务器返回给客户端的数据部分。它通常是在 API 调用、HTTP 请求中，服务器向客户端提供的实际数据（如 JSON、HTML、文件等）。

3.1.1 响应体的特点：

- 内容本身：响应体就是实际数据，而不是函数、接口或数据结构。它包含了根据请求生成的数据。
- 格式化：响应体的内容通常遵循某种标准格式（如 JSON、XML、HTML）。

3.2 LLM 的输出与响应体

在与 LLM 交互时，模型的输出（如生成的文本）可以看作是一个响应体。尽管 LLM 并不直接涉及网络请求，但它生成的文本内容本质上类似于一个响应体。

3.2.1 例子：

请求：用户输入："请告诉我巴黎的天气如何?" LLM 响应体：模型返回的文本："今天巴黎的天气晴朗，气温为 18°C。"

3.3 响应体与后端组件的关系

在 API 调用中，响应体通常是后端逻辑（如函数或服务）执行后生成的内容。LLM 生成的文本响应也是依赖于内部模型组件（如 Transformer 架构、训练数据等）处理输入提示后生成的。

—

4 数据库设计：'BIGINT'、'VARCHAR' 和主键

4.1 'BIGINT' 类型

'BIGINT' 是一种数据库字段类型，用于存储大整数。通常用于存储大范围的数字，如用户 ID、订单号等。

4.1.1 'BIGINT' 范围：

- 有符号：从 -9,223,372,036,854,775,808 到 9,223,372,036,854,775,807。
- 无符号：从 0 到 18,446,744,073,709,551,615。

4.2 'VARCHAR' 类型

'VARCHAR' 是一种变长字符串类型，适用于存储长度可变的文本数据，如用户名、电子邮件、评论内容等。与固定长度的 'CHAR' 类型相比，'VARCHAR' 能有效节省存储空间。

4.3 自增主键与普通主键的区别

- 自增主键：自动生成递增的唯一标识符，常用于需要自动增长的字段，如用户 ID。
- 普通主键：需要手动提供唯一值，并确保每条记录的唯一性。

—

5 Transformer 感受野的讨论

5.1 感受野的定义

** 感受野（Receptive Field）** 是指模型能够感知或依赖的数据范围。在 LSTM 中，感受野是局部的，需要逐步扩展；而在 Transformer 中，由于自注意力机制，每个词都可以直接与其他词进行交互，从而实现全局感受野。

5.1.1 Transformer 的全局感受野：

- 自注意力机制：允许每个位置的元素（如词）与序列中的所有其他元素建立关系，因此具有全局感受野。
- LSTM 的局部感受野：只能逐步扩展，依赖于前一个时间步的状态。

—