光华管理学院 Guanghua School of Management

因思想 而光华

大语言模型与预测
LLMs and Forecasting

李丰

北京大学光华管理学院

https://feng.li/forecasting-with-ai

大模型时代

In the era of LLMs

还有人没有用过大模型吗？

- 你多久会用一次大模型？
- 你用大模型做过预测吗？

Source: https://www.science.org/content/article/ancient-human-sequenced-first-time

LLMs 的发展脉络：以 ChatGPT 为例

- **GPT-1（2018）参数规模：约 1.1 亿**
  - 核心突破：首次把 Transformer 架构 用在语言建模
  - 意义：概念验证了“预训练 + 微调”在 NLP 中可行
- **GPT-2（2019）参数规模：15 亿**
  - 能力提升：能生成较连贯的段落，具备“少样本学习”雏形
  - 影响：因为能生成“以假乱真”的文本，首次让公众意识到生成式 AI 的潜力
- **GPT-3（2020）参数规模：1750 亿**
  - 核心能力：零样本/少样本学习（Few-shot/Zero-shot），可不经微调完成多任务
  - 应用爆发：成为 ChatGPT 的基础，推动 AIGC 概念兴起
  - 局限：幻觉问题严重，可解释性弱，使用成本高
- **GPT-4（2023）参数规模：（估计千亿级以上）**
  - 关键能力：更强的推理与逻辑能力
  - 支持 多模态（文本 + 图片输入）
  - 更稳健的事实性与安全性
  - 应用：广泛进入教育、金融、医疗、企业应用
- **GPT-5（2024–2025）**

  更强的多模态：处理文本、图片、音频、视频
  更接近通用智能（AGI）：跨任务自适应
  更高效：通过小模型组合/量化加速降低算力成本
  企业级应用深化：决策支持、自动化工作流

什么是大语言模型（LLMs）

- **定义：**在海量文本数据上训练的深度学习模型，能够理解和生成自然语言

- **直观类比：**像一个“**有联想功能的超级输入法**”
  - 基于上下文预测下一个词，从而形成智能对话能力
  - 纠正输入错误，猜测你想说什么

- **与传统 AI 的区别：**从“规则驱动/任务专用”转向“通用智能/跨任务”

LLMs 的关键能力

- **语言理解：**阅读、总结、翻译、问答
- **内容生成：**写作、代码、报告、演讲稿
- **推理与逻辑：**链式思维、数学计算、案例分析
- **多模态：**文字 + 图片/音频/视频的处理（如多模态 GPT）
- **人人都可以借助大模型成为专家**

统计模型 vs. 大模型

维度	统计模型	大模型（LLM 等）
目标与哲学	强调解释与推理，关注变量关系	强调预测与模式识别，直接学习复杂规律
假设	依赖明确的分布/结构假设（如线性、正态性）	基本无需假设，端到端学习
可解释性	高，可通过系数、置信区间解释	低，常被视为“黑箱”
数据需求	中小规模数据即可稳健运行	依赖大规模数据与算力
泛化能力	取决于假设正确性，偏离时性能下降	多源、多模态迁移能力强，但可能“幻觉”
应用场景	政策分析、因果推理、实验研究	文本生成、跨模态预测、大规模商业应用
类比	显微镜 → 看清细节与因果	望远镜 → 识别远处复杂模式

LLMs 的局限与挑战 (2025)

- **幻觉问题：**有时会“编造”信息
  - LLM 并不是“查找事实”，而是根据概率分布预测下一个最可能的词。
  - 模型内部没有数据库或事实校验机制，它不会自己去查证答案。
  - 你考试没背住答案时，会不会写一个“一本正经胡说八道”回答?
- **数据隐私与安全：**输入数据是否被泄露
  - 当你使用大模型平台时你的哪些信息被上传了?
- **可解释性不足：**预测/回答的依据不透明
  - 你怎么验证大模型给你的结果?
- **成本与算力：**模型训练和使用的资源开销大
  - 你什么时候有必要本地部署大模型?

大语言模型的标记和向量表示

Token and Embedding

LLMs 中的 Token

- **Token（标记）**：是大语言模型处理的最小输入单位。
- 文本不会直接输入模型，而是先被切分成 token。
- 在文本中Token 可能是一个字、一个词，甚至一个词的一部分。

- 文本例子
  - ChatGPT 是一个强大的模型
  - ["Chat", "G", "PT", "是", "一个", "强大", "的", "模型"]
- 图像例子

LLMs 中的 Token Embedding（向量表示）

- **原始文本:** Forecasting is powerful
- **Token 化:** ["Forecast", "ing", "is", "powerful"]
- **Token ID 映射**
  - "Forecast" → 5021, "ing" → 109, "is" → 42, "powerful" → 7812

- **Token ID 就像词典里的页码。**
- Embedding **把单词的“意义”翻译成数字语言**，让计算机能理解。
- 每个 Token 最终变成一个 连续向量，进入模型进行计算。

Token	ID	Embedding 向量
Forecast	5021	[0.12, -0.85, 0.33, 0.47]
ing	109	[-0.44, 0.29, 0.61, -0.02]
is	42	[0.05, 0.97, -0.12, 0.44]
powerful	7812	[0.88, -0.13, 0.52, 0.09]

基于离散化时间序列的 Tokenize 方法

- **数值直接序列化（Naive Tokenization）**
  - 把每个数值当成一个 token。
  - 例：销量序列 [100, 120, 90] → tokens = ["100", "120", "90"]
  - 简单直观，但不适合数值范围很大或连续性强的场景。
- **区间/等级离散化（Quantization / Binning）**
  - 把数值分桶，映射到有限个 token。
  - 例：销量区间 {0–100, 101–200, 201–300} → ["低", "中", "高"]
  - 优点：减少词表大小，模型更容易学习。
  - 类似文本里的“词汇表”。
- **差分 Token（Delta Encoding）**
  - 把原值转成相邻差值（变化量）。
  - 例：[100, 120, 90] → 差分 = ["+20", "-30"]
  - 更适合捕捉“变化模式”，常用于金融、传感器数据。

Embedding-based Tokenization

- 使用神经网络把每个数值（或子序列）映射成向量。
- 得到的 Embedding 相当于“连续 token”，再输入 LLM。

- **常见模型**
  - **TimeGPT** https://www.nixtla.io/docs/introduction/introduction
  - **Chronos** https://github.com/amazon-science/chronos-forecasting

时间序列的 Tokenize 方法对比

方法	思路	优点	局限
数值直接	原值当 token	简单直观	词表大，难泛化
区间离散	数值分桶	词表小，泛化好	精度损失
差分编码	转换成变化量	捕捉动态趋势	易受噪声影响
SAX 符号化	曲线 → 符号串	类似自然语言	信息压缩较多
滑动窗口	窗口当 token	捕捉局部模式	序列变长
Embedding	映射向量	保持连续性，灵活	需要额外训练

为什么数值型时间序列还需要 Tokenize/Embedding

- **LLM 的输入机制**
  - LLM 天生是为文本序列设计的，它处理的是 token 序列（词或子词），而不是连续数值。
  - 如果直接把原始数值（比如 123.45）输入，模型无法像处理“单词”那样去建模上下文。
- **Tokenize 的作用**
  - **统一格式**：
    - 文本、图像、时间序列都被转成 token 序列
    - LLM 才能用统一的 Transformer 结构处理。
  - **降低复杂度：**把连续值离散化成有限 token，避免模型词表无限大。
  - **捕捉模式：**差分/分桶/窗口化等 tokenization，可以强调趋势或局部结构，而不是孤立数值。

Embedding 的作用

- Embedding 向量能**捕捉语义相似性**：
  - 比如 100 和 101 的 embedding 应该很接近
  - 而 100 和 1000 的 embedding 差距更大

- Embedding 把数值转化为一组**特征维度**
  - **如** 贵/便宜、稳定/波动大等
  - 更利于学习规律
  - 但是不一定是可解释的

- **语言模型**能够理解**数值之间的相对关系**，而不仅仅是“标签”。

1940 是原始数值

Embedding 描述了 1940 的这个值的一组特征：

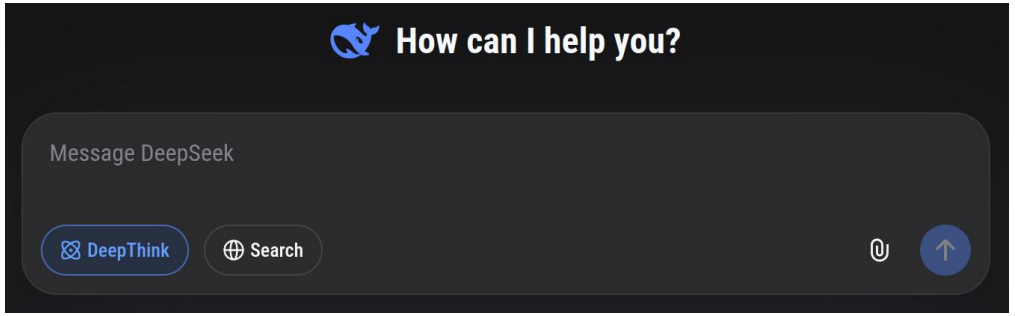
1. 比去年有降价
2. 节假日有周期性
3. 历史最低价
...

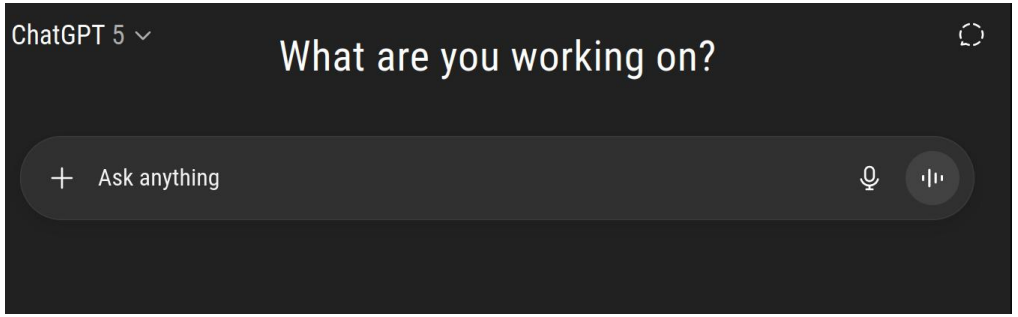
与大模型交互

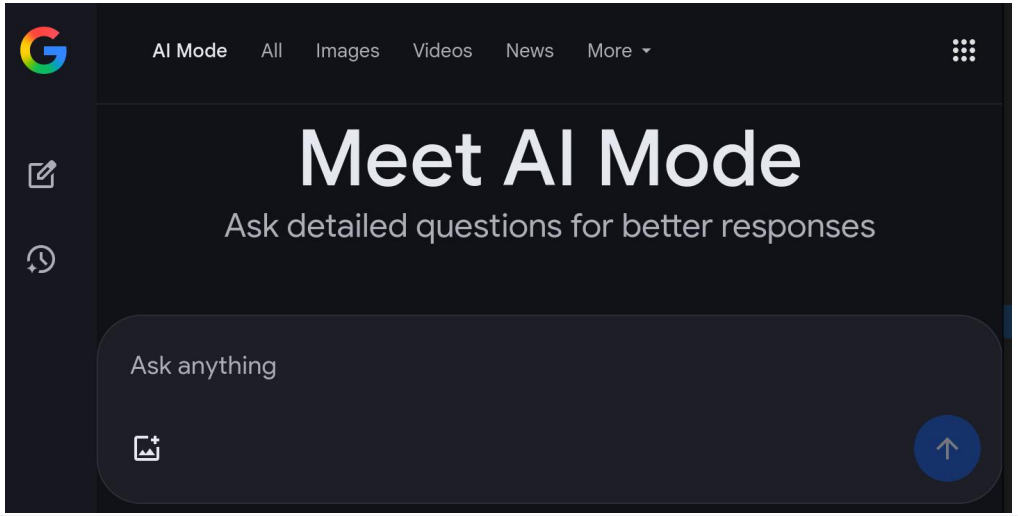
Interaction with LLMs

文本提示（Prompt）

- **Prompt（提示）**：是输入给大语言模型的文本指令或上下文，引导模型生成所需输出。就是和模型对话时你说的第一句话。
- **Prompt 决定了模型该往哪个方向思考**
  - 模型本身没有“目标意识”，它只是根据上下文预测下一个 token。
  - 不同的 Prompt，会让同一个模型给出完全不同的答案。
- **Prompt 的常见类型**
  - **指令型 Prompt：直接给出任务**
    - 例：“请总结以下文章的要点。”
  - **上下文型 Prompt：提供背景信息**
    - 例：“假设你是一位财务分析师，请解释这份报表。”
  - **Few-shot Prompt：给几个示例，让模型模仿**
    - 例：“例子：1+1=2；2+2=4；那么 3+3=？”
  - **Chain-of-thought Prompt：鼓励逐步推理**
    - 例：“请一步一步解释你的推理过程。”







API

- **API（应用程序接口）是开发者调用大模型的接口**
- 它规定了**“怎么把请求发给模型，怎么接收结果”**
  - **自动化：**可以批量调用，适合大规模任务（如成千上万条文本摘要、时间序列预测）。
  - **可集成：**能与企业内部系统结合（CRM、ERP、财务系统），嵌入业务流程。
  - **灵活性高：**可以自定义 Prompt、参数（温度、最大 token）、上下文管理。
  - **扩展性强：**可结合其他数据源、算法，构建定制化 AI 应用。

- **API 和 Prompt**
  - API = 电话线路：保证你能打通对方。
  - Prompt = 你说的话：决定对方如何回应你。

Use the Responses API with Code Interpreter — python

```python
from openai import OpenAI

client = OpenAI()

instructions = """
You are a personal math tutor. When asked a math question,
write and run code using the python tool to answer the question.
"""

resp = client.responses.create(
    model="gpt-4.1",
    tools=[
        {
            "type": "code_interpreter",
            "container": {"type": "auto"}
        }
    ],
    instructions=instructions,
    input="I need to solve the equation 3x + 11 = 14. Can you help me?",
)

print(resp.output)
```

Link an external URL to a file to use in a response — python

```python
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input=[
        {
            "role": "user",
            "content": [
                {
                    "type": "input_text",
                    "text": "Analyze the letter and provide a summary of the key points."
                },
                {
                    "type": "input_file",
                    "file_url": "https://www.berkshirehathaway.com/letters/2024ltr.pdf",
                },
            ],
        },
    ]
)

print(response.output_text)
```

为什么 LLMs 可以做时间序列预测

- **预测任务一致**：时间序列预测和语言建模本质相同，都是“根据已有序列预测下一个元素”
  - **在语言中**：预测下一个词
  - **在时间序列中**：预测下一个数值

- **LLMs 擅长在序列中捕捉上下文依赖**

- **序列化表示：**
  - 时间序列可以转化为“类文本”的输入

为什么要用 LLMs 做时间序列预测

- 时间序列预测在商业与管理中的典型应用场景：零售销量、金融市场、能源需求、旅游预测等
- 传统方法（ARIMA、VAR、机器学习）的局限：特征提取依赖人工、难以适应复杂非线性、多源数据
- LLM 的优势：跨模态、跨任务的强泛化能力，可重编程 (reprogramming) 为时序预测模型

LLMs 与时间序列预测的结合路径

- **直接建模 (LLMs as Forecaster)**
  - 将时间序列转换为文本提示（prompt），LLM 输出未来值
  - 案例：用 GPT 预测销售额
- **间接建模 (LLMs as Feature Extractor/Assistant)**
  - 从新闻、报告、对话中提取影响因子，再与时序模型结合
  - 案例：用 LLM 分析文本情绪来辅助股票预测
- **模型重编程 (Time Series Reprogramming)**
  - 将时序预测任务转化为语言建模问题
  - 案例：将电力负荷曲线转为“句子”输入 LLM
- **混合框架 (Foundation Models for Time Series)**
  - LLM 与专用时序模型（如 Transformer 结构的 TimeGPT、Chronos）融合

案例实践

- Forecasting with DeepSeek Prompt and API