ChatTime-文本时序多模态时序基础模型讲解

1. 论文基本信息 (Bibliographic Information)

- **标题 (Title):** ChatTime: A Unified Multimodal Time Series Foundation Model Bridging Numerical and Textual Data
  - **核心主题：** 论文介绍了一个名为 "ChatTime" 的多模态时序模型。这个模型的核心目标是成为一个“基础模型”，能够同时理解和处理两种完全不同的信息：**时间序列数据**（例如股票价格、天气变化的数值）和**文本数据**（例如新闻、报告）。
- **作者 (Authors):** Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, Jianxin Liao.
  - **研究背景和机构：** 作者主要来自**北京邮电大学 (Beijing University of Posts and Telecommunications)**、**鹏城实验室 (Pengcheng Laboratory)** 以及 **中国联通网络通信有限公司 (China Unicom)**。
- **发表期刊/会议 (Journal/Conference):** The Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI-25)
- **发表年份 (Publication Year):** 2025

2. 整体概括

2.1 研究背景与动机 (Background & Motivation - Why)

时间序列数据构成了现代数字化世界的脉搏。从金融市场的波动曲线到电网的负荷分布，从重症监护室的生命体征监测到全球气候变化的长期趋势，时间序列分析在各个关键行业中扮演着决策支撑的核心角色 。

2.1.1 传统深度学习预测器的局限性

在过去十年中，基于深度学习的时间序列预测模型（如基于RNN、CNN以及Transformer架构的变体）取得了显著的成就。然而，现有的主流方法大多局限于单模态（Unimodal）数值数据。人类专家在分析时间序列时，往往会结合多模态信息。例如，宏观经济学家在预测股市走向时，不仅会观察历史K线图（数值模态），还会深度研读央行的政策报告、行业新闻以及市场情绪分析（文本模态）。然而，传统的Transformer或MLP模型（如DLinear, PatchTST）在设计上天然排斥非数值输入，无法理解和利用这些对预测至关重要的文本上下文信息。

2.1.2 大语言模型带来的范式重构机遇

与此同时，预训练大语言模型（LLM）的爆发式发展为时序分析带来了全新的视角。LLM在海量文本数据上通过自回归预训练，习得了强大的序列建模能力、上下文推理能力以及零样本泛化能力。鉴于语言生成（预测下一个词）与时序预测（预测下一个值）在数学本质上都可归结为对序

列历史依赖关系的建模（即n阶马尔可夫过程），学术界开始探索将LLM迁移至时序领域的可能性。

目前的探索主要分为三条路径：

1. **直接提示法（Direct Prompting）：** 如LLMTIME，将数值序列视为文本字符串（String Representation），直接输入冻结的LLM进行预测 1。这种方法虽然利用了LLM的零样本能力，但由于采用逐位（bit-by-bit）的分词方式，导致Token利用率极低，推理成本高昂，且难以捕捉数值的物理意义。

2. **结构化适配法（Structural Adaptation）：** 如Time-LLM和GPT4TS，通过微调额外的输入输出层来适配数值数据。这种方法在一定程度上保留了LLM的能力，但往往丧失了纯粹的零样本预测能力，且通常只支持数值输入，无法进行多模态交互。

3. **从头预训练法（Training from Scratch）：** 如TimeGPT, Chronos, Moirai，利用海量时序数据从零开始训练类似GPT的架构。这种方法构建了真正的时序基础模型，但代价是极其高昂的训练成本（如Chronos使用了250亿Token，TimesFM使用了3万亿Token），且由于摒弃了原本的文本训练数据，这些模型往往丧失了处理文本信息的能力，沦为单模态工具。

Method	Zero-Shot Forecast	Missing Support	Training Token	Trainable Parameter
TimesFM	✓	✗	3T	200M
Moirai	✓	✓	150B	300M
TimeGPT	✓	✓	100B	Unknown
MOMENT	✗	✓	100B	300M
Timer	✗	✗	50B	50M
Chronos	✓	✓	25B	700M
ChatTime	✓	✓	1B	350M

Table 1: The comparison between pre-trained time series foundation models.

2.1.3 论文的创新点或试图填补的研究空白

因此，ChatTime的想要解决的核心研究空白是：构建一个**统一的模型**，它既能处理**数值**也能处理**文本**，并且像 GPT-4 一样，在不针对新任务进行额外训练的情况下，就能直接对**任意**时间序列进行**零样本预测 (Zero-Shot Forecasting)**

2.2 核心贡献/主要发现 (Main Contribution/Findings - What)

- **论文的主要贡献是什么？**
  1. **提出了 ChatTime 模型：** 这是一个**多模态时间序列基础模型**。
  2. **一个创新的核心思想：** 将时间序列**视为一种“外语”**。通过这种方式，成功地让原本只懂文本的 LLM 理解了数值数据的模式。
  3. **实现了强大的功能：** ChatTime 是一个统一的框架，支持文本和时间序列的双向输入与输出（例如：文本输入 -> 时间序列输出；时间序列输入 -> 文本输出）。
  4. **创建了新的资源：** 建立并发布了**四个新的多模态数据集**，因为之前缺少既有时间序列又有对应文本的数据集，这填补了领域空白，有助于未来的研究。

3、方法 (Methods - 核心技术)

3.1 理论假设：序列建模的统一性：n阶马尔可夫过程

语言模型的核心任务是预测序列中的下一个Token，形式化为计算条件概率 $P(w_t \mid w_{t-n}, \ldots, w_{t-1})$。同样，时间序列预测的核心也是根据历史观测值预测未来值，即计算 $P(x_t \mid x_{t-n}, \ldots, x_{t-1})$ 。ChatTime的作者认为，尽管数据模态不同（离散的符号 vs. 连续的数值），但这两种任务在底层逻辑上都服从**n阶马尔可夫过程（n-order Markov Process）**。

这意味着，LLM在海量文本上习得的Transformer架构——包括多头自注意力机制（Self-Attention）对长距离依赖的捕捉能力、前馈神经网络（FFN）对非线性关系的拟合能力——在理论上是可以直接迁移到时序数据的。关键的挑战在于如何进行**模态对齐（Modality Alignment）**，即如何将连续、无限的数值空间映射到LLM所熟悉的离散、有限的符号空间，同时保留数值的时序特征（趋势、周期性、波动性）。

3.2 词表扩展法：克服“重编程”与“字符串化”的弊端

在ChatTime之前，LLM与时序的结合主要面临两种方法的局限：

1. **字符串表示的低效性（LLMTIME）：** 将数值 0.123 拆解为 0, ., 1, 2, 3 五个Token。这种方式破坏了数值的整体语义，使得模型必须花费大量的注意力去学习“进位制”和“小数点对齐”等基础算术规则，而非专注于时序模式本身。更致命的是，这种方式导致Token消耗极快，极大地压缩了模型能处理的历史窗口长度。

2. **重编程与独立层（Time-LLM）：** 通过训练独立的Embedding层将数值映射到文本的语义空间。虽然解决了效率问题，但这通常需要针对特定数据集进行微调，且往往切断了数值与LLM原有知识的直接联系，使得模型难以进行跨模态的深度推理。

ChatTime创新性地引入了计算语言学中用于多语言适配的**词表扩展（Vocabulary Expansion）**技术。在NLP领域，当需要让一个英语模型掌握韩语或泰语时，研究者通常会扩充Tokenizer的词表，加入新语言的Token，并仅微调Embedding层和输出层，而冻结或轻微调优中间的Transformer层。ChatTime将“时间序列”视为一种新的“外语”，通过量化（Quantization）生成一系列代表不同数值区间的专用Token（如 <Bin_123>），并将这些Token加入LLM的词表。

这种方法的优越性在于：

• **语义完整性：** 每个数值被映射为一个独立的Token，保留了其作为“一个数据点”的完整性，模型无需再进行字符级的拼凑。

• **共享潜在空间：** 新增的时序Token与原有的文本Token共享同一个向量空间。这意味着模型可以像理解“上涨”这个词一样，理解代表数值上升的Token序列。这种深层的语义融合是ChatTime能够实现高质量多模态问答（TSQA）的理论基石。

• **知识复用：** 通过复用预训练的Transformer权重，ChatTime直接继承了LLM对序列模式（如重复、交替、突变）的敏锐感知力，从而避免了从零训练所需的巨大数据量。

3.1 方法原理 (Methodology Principles)

- **核心思想：** 如何让一个只懂 "hello" 和 "world" 的**大型语言模型 (LLM)**，去理解 `[20.5, 21.1, 20.8]` 这样的数字序列?
- **答案：** 把这些数字“翻译”成 LLM 能懂的“外语”。
- **具体原理：** ChatTime 的基础“大脑”是一个预训练好的 LLM (论文中使用的是 **LLaMA-2-7B**)。LLM 的核心是一个“字典”（称为词汇表 Vocabulary）。ChatTime 的方法是**扩展 (Expand)** 这个字典，**添加上万个“新词”**。
- 每一个“新词”就对应一个特定的数值。例如，`0.3529` 这个数字，在“翻译”后变成了 `###0.3529###`。这个 `###0.3529###` 被作为一个**完整的、单独的“单词”** 添加到 LLM 的字典中。
- 这样一来，一个时间序列 `[0.3529, 0.4999, 0.4401]` 就被“翻译”成了一句“外语”：`"###0.3529### ###0.4999### ###0.4401###"`。
- LLM 看到这句“外语”时，它并不知道这是数字，它只知道这是几个它新学会的“单词”。然后，模型被训练去预测这句“外语”的**下一个“单词”**，这个过程与 LLM 预测“今天天气”的下一个词是“真好”完全相同。
- 最后，当 LLM 预测出下一个“单词”（如 `###0.2417###`）时，系统再将其“翻译”回数字 `0.2417`，从而实现了预测。

3.2 方法整体步骤与流程 (Steps & Procedures)

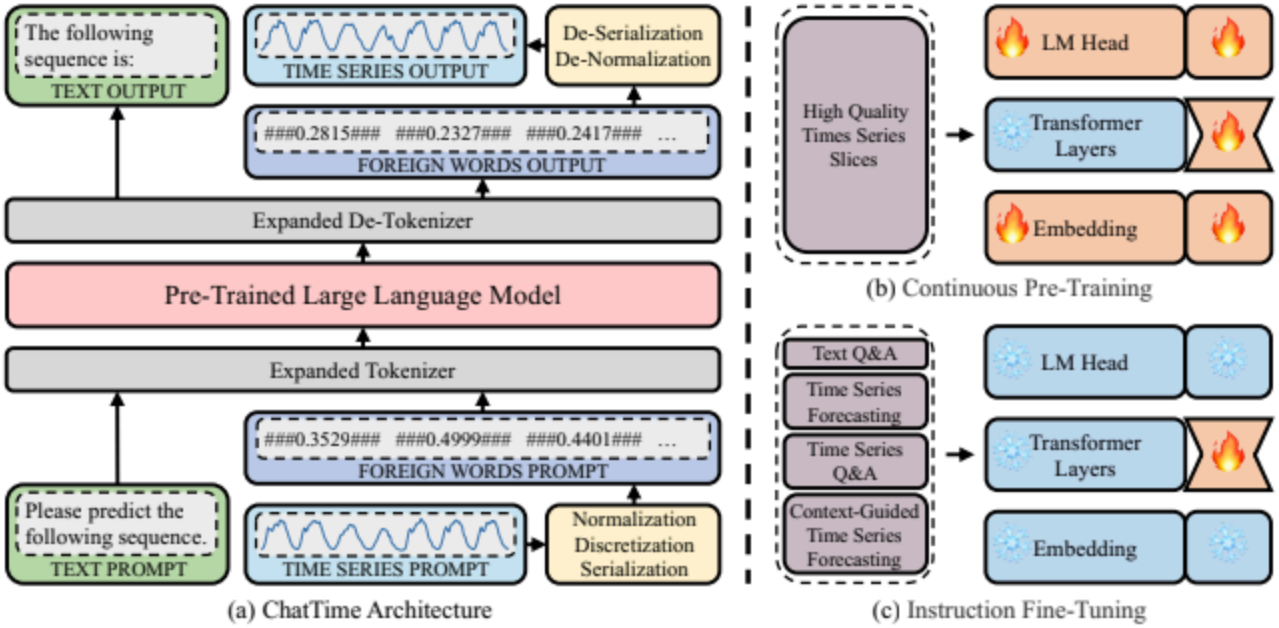

Figure 1: The overview of ChatTime. (a) illustrates the overall architecture, introducing the yellow plug-ins that enable the intertranslation of time-series real values and foreign language. The vocabulary of the grey tokenizer is also extended to accommodate the time series language. We further pre-train (b) and fine-tune (c) existing LLMs using the same methodology as vocabulary expansion, eliminating the need to train from scratch or alter the model architecture.

参见论文图 1(a)，整个流程分为“编码”和“解码”：

1. **输入 (Time Series Prompt):** 获得原始的数值序列 (例如 `[20.5, 21.1, 20.8, ...]`)。
2. **归一化 (Normalization):**
   - **[专业概念解释：归一化]** 原始数据（如股价）可能在 1,000 到 2,000 之间，而其他数据（如温度）可能在 -10 到 30 之间。这种尺度 (Scale) 的巨大差异会“迷惑”AI 模型。
   - **归一化** 就是将所有数据“压缩”到一个**固定的、有界的范围**内（例如 -1 到 1 之间）。论文中使用了一种**最小-最大缩放 (Min-Max Scaling)** 方法，它将历史数据缩放到 **-0.5 到 0.5** 的范围内。
   - *为什么是 -0.5 到 0.5？* 作者特意留出了 `[-1.0, -0.5]` 和 `[0.5, 1.0]` 作为**缓冲区 (Buffer)**，因为未来的预测值很可能**超出**历史数据的最大值或最小值。
3. **离散化 (Discretization):**
   - **[专业概念解释：离散化]** 归一化后，我们得到的仍然是连续的实数（如 `0.451`、`0.4511`、`0.45111...`）。这种无限的精度意味着 LLM 的“字典”需要有无限个“单词”，这是不可能的。
   - **离散化** (也称 **分箱 Binning**) 就是一种“四舍五入”。论文将 -1 到 1 的范围均匀地划分成 **10,000 个“箱子”**。所有落入同一个“箱子”的连续数值（例如 `0.4511` 和 `0.4512`）都被视为*同一个值*（即那个“箱子”的中心点值）。
4. **序列化 (Serialization) / “翻译”:**
   - 这是最关键的一步。将上一步得到的 10,000 个离散值，通过添加标记（如 `###`）的方式，变成 10,000 个唯一的“外语单词”。
   - 例如，`0.2835` 变为 `"###0.2835###"`。

- **(见论文表 2)** 这种方法**极其高效**。对于一个序列 `[0.2835, 0.2285, 0.1587, 0.4001]`，传统方法（如 GPT）可能需要 34 个 词符 (Token) 来表示它（因为它会把 `0`，`.`，`2`，`8` 等拆开）。而 ChatTime **只需要 7 个词符**（4 个“单词”+ 3 个逗号），极大地节省了计算成本。

5. **LLM 处理:** 扩展了词汇表的 LLaMA-2-7B 模型开始处理这句“外语”。
6. **输出 (Foreign Words):** LLM 预测出下一个“外语单词”，例如 `"###0.2417###"`。
7. **反序列化/反归一化 (De-Serialization/De-Normalization):**
   - 系统将 `"###0.2417###"` 翻译回离散值 `0.2417`。
   - 再通过反归一化计算，将 `0.2417` 还原回原始的真实世界尺度（例如 `22.5°C`）。

3.3 方法详解

ChatTime的架构设计精巧地解决了一个核心矛盾：连续无界的数值空间与LLM离散有限的词表空间之间的矛盾。其核心组件，负责在真实数值与“外语单词”之间进行双向转换，以下是方法的详细阐述。

3.3.1 归一化策略：Min-Max Scaling与缓冲区设计**

归一化是时序分析的第一步，旨在消除不同序列量纲（Scale）的差异。然而，ChatTime采用了与竞争对手Chronos截然不同的归一化哲学，这一差异深刻影响了模型的特性。

1. 传统的均值缩放（Chronos方法）

Chronos等模型通常采用均值缩放（Mean Scaling），即将序列除以历史值的绝对平均值：

$$x\_scaled = x / Mean(|x\_history|)$$

这种方法的优点是鲁棒性强，能够保留数据的统计分布特性。但其缺点在于，对于波动较小的序列，归一化后的数值可能聚集在一个极小的范围内，导致模型难以感知局部的微小变化（Local Patterns）。

2. ChatTime的带缓冲区Min-Max缩放

ChatTime采用了**Min-Max Scaling**，但这并非标准的归一化到 。考虑到预测阶段的未来值可能超出历史值的范围（非平稳性），ChatTime设计了一个独特的**缓冲区（Buffer Zone）** 机制。

具体而言，ChatTime将历史序列映射到 $[-0.5, 0.5]$ 的区间内：

$$\tilde{x}_{1:C+H} = \frac{x_{1:C+H} - \min(x_{1:C})}{\max(x_{1:C}) - min(x_{1:C})} - 0.5$$

其中 $x_{1:C}$ 是历史序列。

• **核心区间：** 历史数据被严格限制在 $[-0.5, 0.5]$。

• **缓冲区：** $[-1.0, -0.5)$ 和 $(0.5, 1.0]$ 的区间被预留给未来可能出现的超出历史极值的预测值（Prediction Series）。

设计洞察： 这种策略的核心优势在于它最大化了**局部对比度（Local Contrast）**。通过将历史数据拉伸到固定区间，模型能够极其敏锐地捕捉到波形的形状、峰谷结构和微小趋势。这对于需要精细化描述波形特征的任务（如TSQA中的“描述波动情况”）至关重要，同时也赋予了模型一定的外推能力，允许预测值在受控范围内突破历史极值 。

3.3.2 离散化与量化：从连续到离散

归一化后的数据仍然是连续浮点数，ChatTime通过**分箱（Binning）** 技术将其转化为离散Token。

• **高分辨率分箱：** ChatTime将 $[-1, 1]$ 的区间均匀划分为 **10,000个Bin**。每个归一化后的数值被映射到最近的Bin索引。

• **精度对比：** 相比于Chronos通常使用的约4096个Token，ChatTime的10000个Token提供了更高的数值分辨率（约为 2times10−4）。这种高精度量化有效抑制了“离散化噪声”，使得模型在还原数值时具有更低的量化误差 1。

• **硬映射（Hard Mapping）：** 这种机制建立了一个确定的查找表，彻底消除了LLM在处理字符串数字时常见的“幻觉”问题（例如将 0.99 错误理解为比 1.0 大很多的数，或在算术运算中出错）。

3.3.3 序列化与特殊标记：构建“外语”

为了让LLM能够区分这些数值Token与普通文本Token，ChatTime引入了特殊的标记字符（Mark Characters）。

• **标记格式：** 每个离散化后的数值被包裹在 ### 中，例如 ###0.2835###。这构成了一个独立的“外语单词” 。

• **缺失值处理：** ChatTime显式地引入了 ###Nan### Token来表示缺失值。这与许多模型（如Timer）直接忽略或需要预插值处理不同，ChatTime允许模型“看到”缺失，从而能够学习到缺失本身的模式（例如传感器故障的持续时间特征）。

• **Token效率革命：**

Time Series *[0.2835, 0.2285, 0.1587, 0.4001]*
GPT (34 tokens) *"2 8 3 5 , 2 2 8 5 , 1 5 8 7 , 4 0 0 1"* ['2', '_', '8', '_', '3', '_', '5', '_', '_', '2', '_', '2', '_', '8', '_', '5', '_', '_', '1', '_', '5', '_', '8', '_', '7', '_', '_', '4', '_', '0', '_', '0', '_', '1']
LLaMA (22 tokens) *"2835, 2285, 1587, 4001"* ['2', '8', '3', '5', ',', '_', '2', '2', '8', '5', ',', '_', '1', '5', '8', '7', ',', '_', '4', '0', '0', '1']
ChatTime (7 tokens) *"###0.2835### ###0.2285### ###0.1587### ###0.4001###"* ['###0.2835###', '_', '###0.2285###', '_', '###0.1587###', '_', '###0.4001###']

Table 2: The comparison of token consumption between LLMTIME and ChatTime.

◦ **LLMTIME:** 0.2835 → 0, ., 2, 8, 3, 5 （6个Token）。

◦ ChatTime: 0.2835 → <Token_ID_2835> （1个Token）。 这一设计将Token效率提升了5到10倍。在同样的上下文窗口限制下（例如LLaMA-2的4096长度），ChatTime可以输入比LLMTIME长得多的历史序列，从而捕捉更长周期的依赖关系 1。

4. 模型训练设置

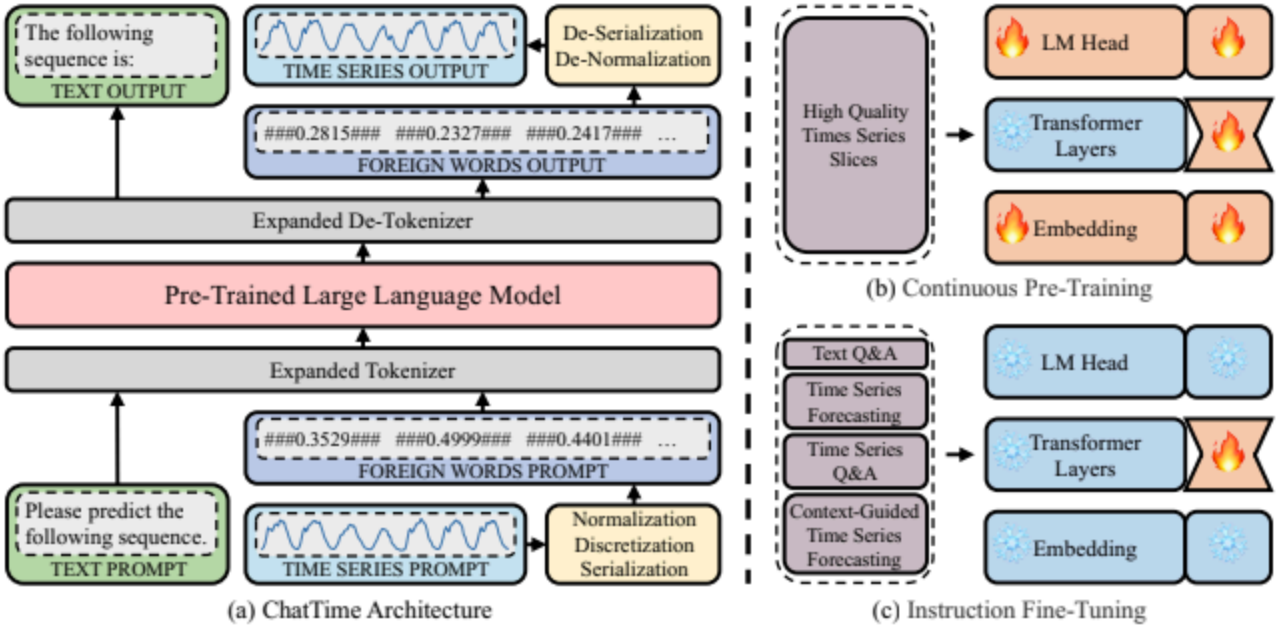

Figure 1: The overview of ChatTime. (a) illustrates the overall architecture, introducing the yellow plug-ins that enable the intertranslation of time-series real values and foreign language. The vocabulary of the grey tokenizer is also extended to accommodate the time series language. We further pre-train (b) and fine-tune (c) existing LLMs using the same methodology as vocabulary expansion, eliminating the need to train from scratch or alter the model architecture.

4.1 前置基础知识

在解释训练细节之前，需要首先明确大模型（如图中的红色/蓝色方块）各层的作用。

1. **Embedding（嵌入层）—— 词典表：**

◦ **功能：** 它的作用是将输入的“单词”转化成计算机能理解的“坐标（向量）”。就像把一本字典里的文字对应到立体的数学坐标系中。

2. **Transformer Layers（特征提取层）—— 核心大脑：**

◦ **功能：** 这是模型最厚、参数最多的部分。它负责理解上下文逻辑（比如：“今天”和“下雨”在一起是什么意思）。在图中，这部分占了模型体积的绝大部分。

3. **LM Head（输出头）—— 发言人：**

◦ **功能：** 它的任务是根据大脑处理的结果，从词典里选出最可能的一个“下一个词”。

4. **LoRA(Low-Rank Adaptation)：** 这是一种 高效微调 技术。LLaMA 有 70 亿个参数，训练全部参数非常昂贵。LoRA 会“冻结”绝大多数的原始参数，只在模型中添加一些*非常小*的“适配器”层进行训练。这极大地降低了训练所需的计算资源（GPU 显存）。

4.2 基础模型与适配器

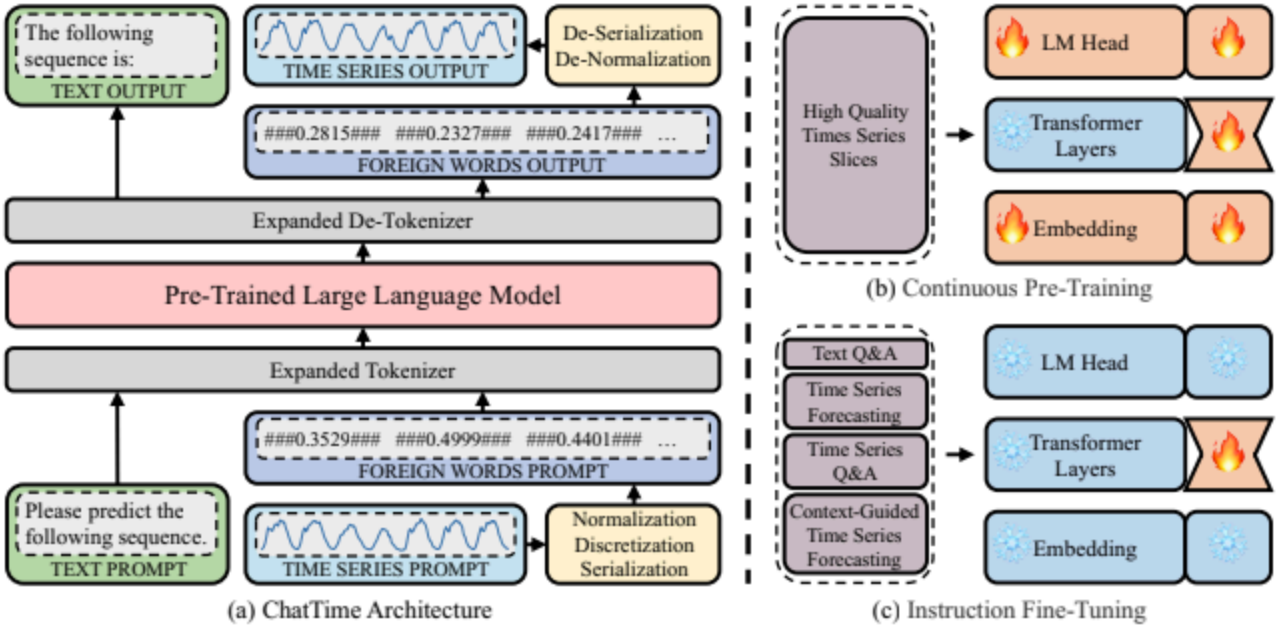

Figure 1: The overview of ChatTime. (a) illustrates the overall architecture, introducing the yellow plug-ins that enable the intertranslation of time-series real values and foreign language. The vocabulary of the grey tokenizer is also extended to accommodate the time series language. We further pre-train (b) and fine-tune (c) existing LLMs using the same methodology as vocabulary expansion, eliminating the need to train from scratch or alter the model architecture.

ChatTime选择 **LLaMA-2-7B-Base** 作为骨干网络。训练过程中，模型并未对所有参数进行全量更新，而是采用了参数高效微调（PEFT）策略：

• **Embedding层与LM Head：** 由于词表扩充了约10001个新Token，输入嵌入层和输出层必须进行调整和训练。

• **Transformer层：** 使用 **LoRA (Low-Rank Adaptation)** 进行微调。Rank设为8，Alpha设为16。这种配置在保证模型适应新模态的同时，最大程度保留了预训练的通用知识，且显著降低了显存需求，使得整个训练过程可在单张RTX 4090上完成 。

---

5. 模型训练详细过程：从数据清洗到指令对齐

ChatTime 的训练**不是从零开始的**，它利用了 LLaMA-2 的强大基础，分两阶段进行：**持续预训练（Continuous Pre-Training, CPT）** 和 **指令微调（Instruction Fine-Tuning, IFT）**。这一流程

复刻了LLM从“语言建模”到“指令遵循”的经典学习路径。

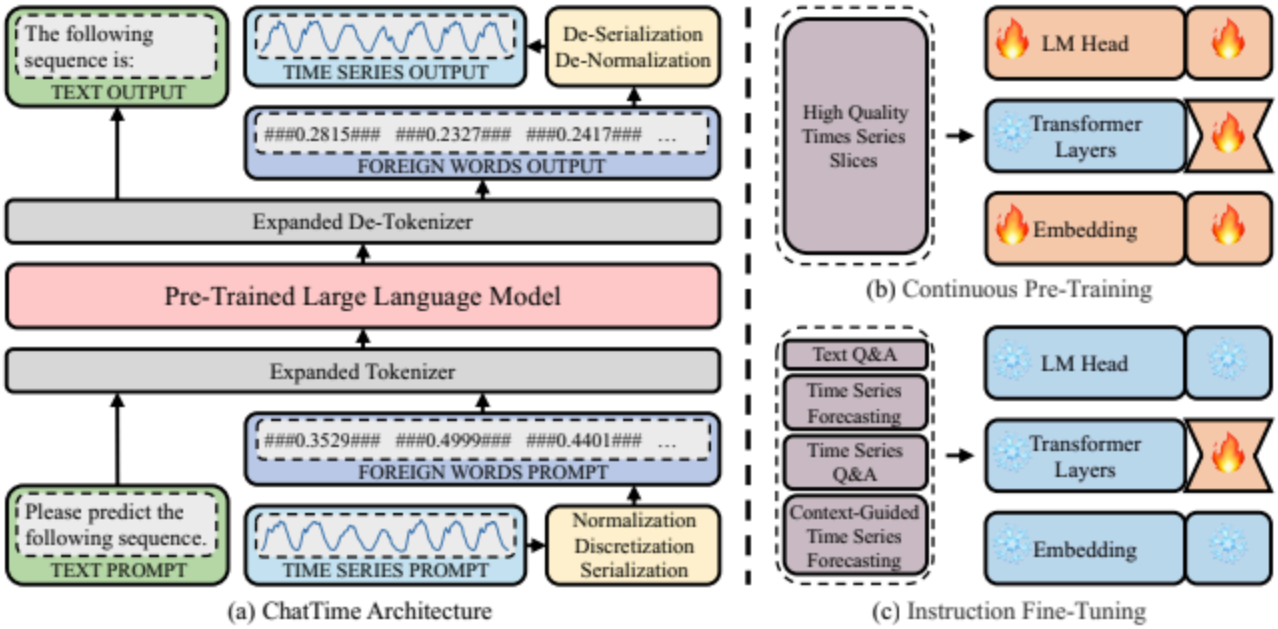

Figure 1: The overview of ChatTime. (a) illustrates the overall architecture, introducing the yellow plug-ins that enable the intertranslation of time-series real values and foreign language. The vocabulary of the grey tokenizer is also extended to accommodate the time series language. We further pre-train (b) and fine-tune (c) existing LLMs using the same methodology as vocabulary expansion, eliminating the need to train from scratch or alter the model architecture.

5.1 持续预训练：习得时序语言的语法

在第一阶段，目标是让模型理解时序数据的内在规律（即时序语言的“语法”）。训练Embedding层和 LM Head层，冻结核心大脑Transformer层，也就是大模型原本的知识库没有在这个阶段被改动。因为这是模型第一次接触这 10,000 个“新词”，它必须在“词典（Embedding）”里给这些词分配新的数学坐标，并在“发言人（LM Head）”那里学会如何正确输出这些新词，因此需要对Embedding层和LM Head层进行训练。

5.1.1 数据集构建与聚类策略

为了避免从头训练的高昂成本，ChatTime并没有使用像Chronos那样庞大的千亿级数据集，而是通过精巧的数据筛选策略实现了“少即是多”。

• **数据源：** 使用 100 万个高质量、多样化的时间序列片段（来自 **Monash** 和 **TFB** 两个大型数据库） 。

• **任务： 自回归预测 (Autoregressive forecasting)**，即“预测下一个值”。

• **滑动窗口切片：** 使用不同大小的窗口（36至576步长）对原始序列进行切片，以捕捉不同频率的模式 。

• **K-Means聚类筛选（关键创新）：** 原始切片数量高达1000万（10M）。为了提炼高质量、高代表性的数据，作者对这1000万个切片执行了 **K-Means聚类**。

◦ **筛选机制：** 将切片聚类为 **100万（1M）** 个簇，并从每个簇中随机选取一个样本作为代表1。

◦ **深度洞察：** 真实世界的时序数据充斥着大量重复、平庸的模式（如长时间的平稳直线或简单的正弦波）。通过聚类，ChatTime剔除了大量冗余数据，强迫模型专注于学习最具代表性和多样性的**波形结构（Shapes）**。这不仅将训练数据量减少了90%，更提高了数据的“信息密度”，是ChatTime能够高效训练的关键。

5.1.2 训练目标

模型在扩展后的词表上执行标准的**自回归预测任务（Next Token Prediction）**。即给定历史Token序列，预测下一个数值Token。由于新Token与旧Token共享参数，模型在这一阶段主要学习的是数值Token之间的转移概率。

5.2 指令微调：多模态对齐与任务泛化

由于第一阶段中模型已经学会了数字的“含义”，因此在微调阶段就不需要再改动基础词典了。第二阶段旨在赋予模型理解指令和处理多模态任务的能力。目的是 教会模型如何*根据人类的指令*（即文本提示词）来完成特定任务，使其从一个“预测器”变成一个有用的“助手”。

5.2.1 数据集构建

微调数据集共包含10万（100K）条样本，由四个任务组成，每个任务贡献2.5万条数据：

1. **单模态时序预测（Unimodal Forecasting）：** 基础的“看图说话”，输入历史数值，输出未来数值。使用聚类筛选出的25K高质量样本。纯数值预测，巩固自回归预测能力。

2. **文本问答（Text QA）：** 引入 **Alpaca** 数据集，确保模型**不会忘记**其原有的文本聊天能力。

◦ **深度洞察：** 这一步至关重要。如果在微调阶段仅使用时序数据，模型可能会发生“灾难性遗忘”，丧失处理纯文本的能力。保留Text QA任务是为了维持LLM的语言理解中枢，使其能处理复杂的Prompt指令。

3. **上下文指导的时间序列预测 (CGTSF): 多模态核心任务**。例如：(输入：[文本-“天气预报显示明天有暴雨”] + [序列-“过去几天的太阳能发电量”]) -> (输出：[序列-“预测明天的发电量”])。引入MSPG（太阳能）等数据集，训练模型提取“暴雨”、“节假日”等文本线索。

4. **时序问答（TSQA）： 时间序列问答 (TSQA): 多模态核心任务**。例如：(输入：[序列-“某股票过去一年的价格”] + [文本-“这段时间的波动性大吗？”]) -> (输出：[文本-“是的，波动性很大”])。针对Trend（趋势）、Volatility（波动）等特征提问，建立数值到语义的映射。

训练配置： 4-bit量化加载模型，LoRA微调，Epoch设为4。这种轻量级的配置证明了基于Foundation Model进行模态扩展的高效性。

---

6. 实验结果与深度分析

ChatTime在三个核心任务上与TimeGPT, Chronos, Moirai等顶尖模型进行了全面对比。

6.1 数据集 (Datasets)

- **训练数据集:**
  - **Monash 和 TFB 数据库：** 用于**阶段一（预训练）**。这是两个大型的公开时间序列数据库，包含了成百上千个来自不同领域（金融、天气、工业等）的数据集。
  - **Alpaca 数据集：** 用于**阶段二（指令微调）**，提供通用的文本问答能力。
- **评估 (实验) 数据集:**
  - **ZSTSF (零样本预测):** 使用了 8 个非常著名的基准数据集 (例如 ETT, Electric, Traffic, Weather)。**重点：** 这些数据集**已从训练数据中被刻意排除**，以确保模型在评估时对它们是完全“陌生”的，从而严格测试其**零样本**能力。
  - **CGTSF (上下文指导预测):** 使用了 3 个作者**自己创建的新数据集** (因为之前没有合适的多模态数据集)：
    1. **MSPG (墨尔本太阳能发电):** 包含太阳能发电量 + 背景、天气、日期信息。
    2. **LEU (伦敦电力使用):** 包含电力使用量 + 背景、天气、日期信息。
    3. **PTF (巴黎交通流量):** 包含交通流量 + 背景、天气、日期信息。
  - **TSQA (时间序列问答):** 使用了作者合成的问答数据集。
- **为什么选择这些数据集?**
  - ZSTSF 的 8 个数据集是该领域的**“高考题”**，用于和所有其他模型进行公平对比。
  - CGTSF 的 3 个新数据集是**“应用题”**，用于验证模型的核心创新点——即**利用文本信息来辅助数值预测**的能力。

6.2 评估指标 (Evaluation Metrics)

- **用于 ZSTSF 和 CGTSF (预测任务):**
  - **MAE (Mean Absolute Error, 平均绝对误差)**
  - **[专业概念解释：MAE]** 这是衡量预测准确度最常用、最直观的指标之一。它计算的是“每个时间点上的**预测值**”和“**真实值**”之间**差距的绝对值**，然后再取所有时间点的平均值。
  - **MAE 越低，说明模型的预测越精准，性能越好。**
- **用于 TSQA (问答任务):**
  - **Acc (Accuracy, 准确率)**
  - **[专业概念解释：Acc]** 这很好理解，就是模型回答正确的**问题百分比**。例如，给模型 100 个关于时间序列特征的问题，它答对了 76 个，那么 Acc 就是 76%。
  - **Acc 越高，说明模型对时间序列特征的“理解”越好，性能越好。**

6.3 实验结果 (Results - 数据解读与结果分析)

6.3.1 实验1：零样本时序预测（ZSTSF）- 见论文表 4

实验内容： 在 8 个标准数据集上测试 ChatTime 的零样本预测能力，并与 SOTA（最先进的）模型对比。

• 具体数据（表4）：

Dataset	Hist	Pred	Full-Shot Forecast				Zero-Shot Forecast				
			DLinear	iTransformer	GPT4TS	TimeLLM	TimeGPT	Moirai	TimesFM	Chronos	ChatTime
ETTh1	48	24	0.1462	0.1650	**0.1389**	0.1467	**0.1604**	0.1694	0.2021	0.1634	0.1698
	72	24	**0.1358**	0.1852	0.1469	0.1439	0.1603	0.1796	0.1599	**0.1372**	0.1403
	96	24	**0.1398**	0.1964	0.1447	0.1473	0.1577	0.1433	0.1454	**0.1374**	0.1374
	120	24	**0.1371**	0.1971	0.1414	0.1513	0.1594	0.1492	0.1502	**0.1348**	0.1431
ETTh2	48	24	**0.2724**	0.2937	0.2742	0.2758	**0.2874**	0.2963	0.3360	0.3128	0.2906
	72	24	0.2756	0.3118	**0.2717**	0.2972	0.2888	0.3109	**0.2880**	0.3045	0.3092
	96	24	**0.2831**	0.3417	0.2900	0.2864	**0.2902**	0.3139	0.3144	0.3158	0.2917
	120	24	0.2863	0.3299	**0.2854**	0.3175	0.3026	**0.2905**	0.3311	0.3150	0.3124
ETTm1	192	96	0.1479	0.1608	**0.1384**	0.1503	0.1921	0.1608	0.1719	0.1604	**0.1442**
	288	96	0.1400	0.1813	**0.1345**	0.1425	0.1715	0.1848	0.1650	**0.1452**	0.1587
	384	96	**0.1428**	0.1680	0.1518	0.1452	0.1616	0.1619	0.1584	0.1463	**0.1393**
	480	96	**0.1406**	0.2001	0.1472	0.1527	0.1570	0.1703	0.1582	**0.1401**	0.1802
ETTm2	192	96	0.2793	0.3397	**0.2792**	0.2918	0.4294	0.4206	0.3405	0.3759	**0.3135**
	288	96	**0.2881**	0.3623	0.2918	0.2904	0.3625	0.3882	**0.3277**	0.3472	0.3340
	384	96	0.2947	**0.2880**	0.3089	0.3003	**0.3389**	0.3742	0.3562	0.3589	0.3434
	480	96	0.3014	0.3725	**0.2945**	0.3054	**0.3242**	0.3597	0.3679	0.3353	0.4213
Electric	48	24	0.5719	0.5951	**0.5008**	0.5733	**0.5276**	0.6617	0.6005	0.6098	0.6083
	72	24	0.5486	0.5619	**0.4896**	0.4989	**0.4953**	0.6018	0.5454	0.5914	0.6238
	96	24	0.5536	0.5290	**0.4432**	0.4816	0.4971	0.5260	0.5276	0.5139	**0.4951**
	120	24	0.4714	0.5622	**0.4540**	0.4848	0.5196	0.4963	**0.4900**	0.5031	0.5101
Exchange	14	7	0.0543	**0.0526**	0.0533	0.0531	0.0620	0.0784	0.0647	0.0555	**0.0540**
	21	7	0.0571	0.0547	**0.0505**	0.0505	0.0599	0.0812	0.0743	0.0635	**0.0556**
	28	7	0.0595	0.0581	**0.0508**	0.0511	0.0610	0.0844	0.0652	0.0595	**0.0559**
	35	7	0.0615	0.0607	**0.0493**	0.0524	0.0629	0.0677	0.0632	0.0598	**0.0558**
Traffic	48	24	0.4662	0.5000	0.4557	**0.4473**	0.4668	0.4887	0.4483	0.4718	**0.4220**
	72	24	0.4475	0.4443	**0.4116**	0.4252	0.4635	0.4581	0.4196	**0.3725**	0.3873
	96	24	0.4438	0.4348	0.4190	**0.4064**	0.4332	0.4082	**0.3714**	0.3787	0.4074
	120	24	0.4190	0.4149	**0.3416**	0.4279	0.4161	**0.3539**	0.3542	0.3908	0.4125
Weather	288	144	**0.0339**	0.0367	0.0364	0.0352	0.0331	**0.0305**	0.0354	0.0343	0.0352
	432	144	**0.0366**	0.0404	0.0401	0.0395	0.0321	0.0302	**0.0298**	0.0346	0.0356
	576	144	**0.0364**	0.0379	0.0399	0.0377	0.0328	0.0331	0.0321	0.0349	**0.0284**
	720	144	**0.0371**	0.0395	0.0392	0.0392	**0.0323**	0.0353	0.0369	0.0335	0.0332
Avg. MAE			0.2409	0.2661	**0.2286**	0.2390	0.2544	0.2659	0.2541	**0.2512**	0.2515
Avg. Rank			3.7500	6.9688	**3.0000**	3.9688	5.5625	6.5000	5.7500	4.8438	**4.4688**

Table 4: The evaluation result in the traditional unimodal time series forecasting task. The lower values for all metrics represent the better performance. The best results among full-shot and zero-shot forecasting methods are highlighted in bold, respectively.

• 结果解读：

- ChatTime (平均 MAE 0.2515) 的表现与当前最强的零样本模型 Chronos (平均 MAE 0.2512) **几乎持平** (达到了 Chronos 99.9% 的性能)。
- **带来的启发 (关键结论)：**
  - **效率对比：**
    - Chronos使用了250亿（25B）Token进行训练，参数量700M；ChatTime仅使用了10亿（1B）Token，参数量350M（可训练部分）。ChatTime用 **4%的数据量** 达成了同等性能。

- **对比全量微调模型：** ChatTime的零样本性能达到了特定数据集全量训练模型（如GPT4TS）的 **90.9%**。这表明，基础模型的通用泛化能力已经非常接近特定领域专精模型的天花板。

深度洞察： 这一结果有力地证明了“词表扩展”策略的有效性。ChatTime无需像Chronos那样从零学习“什么是序列”，而是激活了LLaMA内部早已存在的深层序列推理能力。LLM在预训练中学到的逻辑推理、因果判断模式，被成功迁移到了数值域。

6.3.2 实验2：上下文指导的时间序列预测 (CGTSF) - 见论文表 5

• 实验内容： 测试模型能否利用**文本上下文**来做出更准的**数值预测**。

• 具体数据（表5）：

Dataset	Hist	Pred	Dataset-Specific Forecast				Dataset-Shared Forecast				
			DLinear	GPT4TS	TimeLLM	TGForecaster	Moirai	TimesFM	Chronos	ChatTime-	ChatTime
MSPG	192	96	**0.7136**	0.7558	0.7697	0.7595	0.8108	0.8362	0.7427	0.7606	**0.7346**
	288	96	**0.7083**	0.7464	0.7959	0.7610	0.7849	0.7896	0.7408	0.7606	**0.7353**
	384	96	**0.7014**	0.7388	0.7672	0.7638	0.7749	0.7811	0.7352	0.7607	**0.7330**
	480	96	**0.7018**	0.7311	0.7632	0.7695	0.7664	0.7667	0.7344	0.7607	**0.7292**
LEU	96	48	0.6676	0.6697	0.6531	**0.6181**	**0.6228**	0.6670	0.6571	0.6496	0.6305
	144	48	0.6495	0.6567	0.6474	**0.6355**	**0.6085**	0.6475	0.6597	0.6506	0.6231
	192	48	0.6407	0.6771	**0.6329**	0.6458	**0.6008**	0.6490	0.6645	0.6407	0.6111
	240	48	**0.6316**	0.6383	0.6356	0.6329	**0.5968**	0.6333	0.6631	0.6377	0.6085
PTF	48	24	0.5204	0.4373	**0.4211**	0.4411	0.5981	0.4851	**0.4813**	0.5155	0.4849
	72	24	0.5075	0.4253	0.4031	**0.3943**	0.5776	**0.4258**	0.4276	0.4436	0.4307
	96	24	0.4965	0.3921	0.4392	**0.3653**	0.5179	0.4054	0.4336	0.4172	**0.3920**
	120	24	0.4796	0.3713	0.3594	**0.3594**	0.5245	0.3807	0.3902	0.3943	**0.3480**
Avg. MAE			0.6182	0.6033	0.6073	**0.5955**	0.6487	0.6223	0.6109	0.6160	**0.5884**
Avg. Rank			4.7500	5.0833	4.9167	**3.9167**	5.9167	6.4167	5.5000	5.7500	**2.5833**

Table 5: The evaluation result in the context-guided time series forecasting task. The lower values for all metrics represent the better performance. The best results among dataset-specific and dataset-shared methods are highlighted in bold, respectively.

• 对比基线：

◦ **ChatTime-：** 屏蔽文本输入，仅看数值。

◦ **TGForecaster：** 专为文本引导设计的竞争模型。

• 结果解读：

◦ ChatTime（MAE 是 0.5884）显著优于ChatTime-（MAE 是 0.6160），证明模型确实“读懂”了文本。例如，当文本提示“明天是公共假期”时，模型能准确预测出交通流量的下降，而纯数值模型只能基于历史趋势盲目外推。

◦ ChatTime在平均Rank上优于TGForecaster（2.58 vs 3.92）。这说明统一的基础模型比针对特定任务设计的单一模型具有更强的语义理解与融合能力。

- **带来的启发 (关键结论)：**
  - **0.5884 远低于 0.6160**。这表明，当 ChatTime 获得了文本信息后，它的**预测误差显著降低了**。

- **结论：这证明了 ChatTime 成功地学会了如何理解文本，并利用这些信息来改进其对时间序列的预测**。这验证了其**多模态**设计的有效性。

6.3.3 实验 3：时间序列问答 (TSQA) - 见论文表 6

- **实验内容：** 给模型一段时间序列，然后用**文本**问它关于这段序列特征的问题（例如："它有明显的趋势吗？" "有异常点吗？"）。这是对模型"数值→文本"翻译能力的终极测试。数据集包含趋势、季节性、波动性、异常值四类特征的问答。
- **具体数据（表6）：**

Feat	Len	GPT4	GPT3.5	GLM4	LLaMA3	ChatTime
Trend	64	0.6532	0.3507	0.7319	0.6799	**0.9011**
	128	0.7015	0.5846	0.7574	0.5855	**0.9068**
	256	0.7482	0.5028	0.6377	0.6143	**0.8843**
	512	0.6346	0.5903	0.6697	0.6753	**0.8234**
Volatility	64	0.5585	0.5633	0.6797	0.6373	**0.7874**
	128	0.4979	0.3839	0.4770	0.4756	**0.6954**
	256	0.4624	0.4894	0.5418	0.5246	**0.6228**
	512	0.3169	0.3796	0.4549	0.5261	**0.5736**
Season	64	0.3518	0.3428	0.3366	0.3484	**0.6639**
	128	0.3515	0.3952	0.3464	0.3958	**0.6517**
	256	0.5283	0.5089	0.3892	0.4120	**0.6463**
	512	0.4457	0.4889	0.3892	0.4127	**0.6244**
Outlier	64	0.7230	0.4325	0.5359	0.7051	**0.8773**
	128	0.6327	0.5940	0.5298	0.5694	**0.9032**
	256	0.6795	0.4579	0.5019	0.5073	**0.8593**
	512	0.6219	0.4996	0.2822	0.4085	**0.7478**
Avg. Acc		0.5567	0.4728	0.5163	0.5299	**0.7605**
Avg. Rank		3.0625	4.0625	3.6250	3.2500	**1.0000**

Table 6: The evaluation result in the time series question answering task. Higher values mean better performance for all metrics, except Rank, which is better when lower. The best results are highlighted in bold.

- **结果解读：**
  - ChatTime的平均准确率（Avg. Acc）高达 **0.7605**。
  - 作为对比，通用LLM表现惨淡：**GPT-4 (0.5567)**, **LLaMA-3 (0.5299)**。

• **带来的启发 (关键结论)：**

◦ 通用LLM将数字视为文本字符串，难以在大脑中构建出“波形”的视觉或几何概念。对于GPT-4来说，0.1, 0.2, 0.3 是一串字符，而不是一条斜率为正的线。

◦ ChatTime通过量化和共享嵌入空间，将数值序列转化为了一种模型可直接感知的“语义流”。它能像理解句子中的语法结构一样，识别出数值序列中的“趋势结构”或“周期结构”。

**6.3.4 消融实验：实验 4：消融研究 (Ablation Study) - 见论文图 2

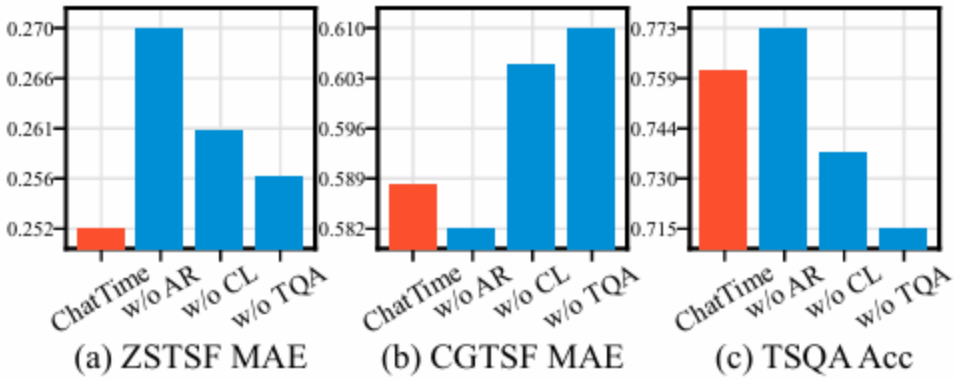

Figure 2: The evaluation result between ChatTime and variants. Lower values are better for ZSTSF and CGTSF, while higher values are better for TSQA.

• **w/o AR（移除阶段一的自回归预训练）：** ZSTSF（零样本预测）任务的 MAE 误差**急剧上升**。这表明直接用少量指令数据微调LLM是行不通的，必须先通过大规模预训练让模型“学会”时序语言。
- **结论：** 阶段一的预训练**至关重要**，它是模型学会零样本泛化能力的**基础**。

• **w/o CL（移除对训练数据的聚类）：** 使用随机采样代替聚类筛选数据，所有任务的性能都变差了。
- **结论：** 这证实了K-Means聚类在提高数据多样性和训练效率方面的关键作用。使用高质量、多样化的数据（通过聚类筛选）进行训练，比随机抓取数据要**好得多**。

• **w/o TQA（移除指令微调中的文本问答数据）：** 所有任务性能都下降，尤其是多模态任务 (CGTSF 和 TSQA)。
- **结论：** 确保模型“不忘本”（保持其文本理解能力）对于其执行**多模态**任务是**必需的**。

---

7. 竞品深度对标：ChatTime vs. Chronos vs. LLMTIME

为了更清晰地定位ChatTime的技术生态位，我们将其与当前最主流的两种范式进行多维度对比。

维度	ChatTime	Chronos (Amazon)	LLMTIME
核心理念	**词表扩展 (Vocabulary Expansion)**	**分类回归 (Regression via Classification)**	**字符串提示 (String Prompting)**
数据形态	扩展的新Token (Foreign Words)	复用现有Token ID	纯文本字符序列
归一化	**Min-Max** (关注局部形状/对比度)	**Mean Scaling** (关注整体分布)	无/简单归一化
分箱精度	**10,000 Bins** (高精度)	~4,096 Bins (中等精度)	依赖文本字符 (低精度/幻觉风险)
Token效率	**极高** (1值=1Token)	**极高** (1值=1Token)	**极低** (1值=3~10 Tokens)
模态支持	**原生双模态 (数值+文本)**	主要是单模态 (数值)	伪多模态 (文本提示)
文本推理	**保留** (共享Embedding空间)	**破坏** (Token ID被挪用，语义混淆)	**保留** (但在长序列下窗口溢出)
缺失值	###Nan### 显式学习	PAD 填充或插值	需预处理插值
最佳场景	需要**解释分析**、多模态交互、复杂指令遵循的场景	大规模、纯粹的**数值预测**任务	简单、短序列的零样本验证

关键结论： Chronos更像是一个极致的“计算器”，它牺牲了语言能力以换取在纯数值预测上的暴力美学；ChatTime则是一个“分析师”，它保留了语言的智慧，能够理解数据背后的语境并进行沟通。LLMTIME则更像是一个过渡性的尝试，受限于Token效率，难以处理长序列实际问题。

---

8. 未来展望：通往全能时序智能体之路

ChatTime虽然取得了突破，但正如其作者所言，它尚未达到能力的饱和点。基于现有的架构优势，未来的研究可以向以下几个战略方向拓展。

8.1 从“点”到“块”：Patching技术的引入

目前ChatTime采用Point-wise Tokenization（1个值=1个Token）。对于高频数据（如每秒采样的传感器数据），序列长度依然会迅速爆炸。

• **优化方向：** 引入 **Patching** 技术，将相邻的（例如16个）数值打包为一个Token（Patch）。这相当于让模型从识“字”进化到识“词组”。

• **预期收益：** 这将使上下文窗口的有效信息容量提升一个数量级，使模型能够捕捉极长周期的依赖关系（如年度季节性），同时大幅降低推理计算量 1。

8.2 认知升级：思维链（Chain of Thought, CoT）

目前的预测是直接输出结果。未来可以引入CoT机制，强制模型在预测前生成推理过程。

• 场景示例：

◦ *Prompt:* “预测明日电力负荷。”

◦ *CoT Output:* “观测到明日气温骤降10度（外部变量），且为周六（日历特征）。推理：工业用电将减少，但居民采暖需求将激增。综合判断：总负荷将上升5%。”rightarrow *Prediction: [数值]*。

• 价值： 这将极大地提升模型的可解释性（Explainability）和在复杂反直觉场景下的鲁棒性 1。

8.3 领域垂直化：打造行业专家

ChatTime目前的训练数据主要是通用的Monash/TFB。

• 金融方向： 结合K线数据与财报、美联储会议纪要、Twitter舆情进行微调，打造能读懂“叙事经济学”的金融预测Agent。

• 医疗方向： 结合ECG/EEG波形数据与医生临床诊断报告，训练能自动撰写病理分析报告的辅助诊断模型。

• 异常检测： 不仅识别“哪里”异常，还能用自然语言解释“为什么”异常（例如：“检测到夜间流量异常归零，推测为数据中心断电”）1。

8.4 多变量协同预测

目前ChatTime主要处理单变量序列。未来可以将不同变量视为对话中的不同“发言者”或并行的“语流”，利用Attention机制捕捉变量间的因果滞后关系（例如：广告投入增加rightarrow 流量增加rightarrow 销量增加），从而实现复杂系统的系统性预测 1。

9. 结语

ChatTime 的提出标志着时序分析领域的一个重要转折点。它证明了不需要为了时序任务而抛弃LLM的语言能力，也不需要设计全新的异构网络。通过 **“时序即外语”** 这一优雅的归约，ChatTime成功地将数值世界的精确性与文本世界的语义丰富性融合在了同一个Transformer架构中。

这项工作不仅提供了一个高效、高性能的零样本预测工具，更为通用的**时序智能体（Time Series Agent）** 奠定了基石。未来的AI将不再仅仅是冷冰冰的数字预测机器，而是能够看懂图表、读懂报告、并能与人类分析师进行深层对话的数据伙伴。在迈向通用人工智能（AGI）的征途中，ChatTime为如何处理非文本模态数据提供了一个极具参考价值的范本。