CSE模型：上下文语义标记的统一建模框架

Contextual Structure Expression: A Unified Modeling Framework for Semantic Contextual Markup

摘要（Abstract）

随着大模型（LLM）的发展，**知识取代信息**逐渐成为核心处理对象，结构化表达的重心也在悄然转移。

以 JSON、YAML、RDF 等为代表的传统结构化表达方式，受限于静态结构与确定性假设，难以应对语义驱动的时代需要。在智能系统中，实际语义常常依赖于背景上下文的变化，预设schema的方式，在处理**动态变化的上下文**时显得极为僵硬。并且由于LLM更擅长处理自然语言，采用JSON、YAML、RDF来交换语义，会导致和自然语言之间的频繁互转，在线性化的过程中，天然会损耗语义，最终带来上下文割裂，导致多模态协同、动态语境绑定、可解释性等方面的各种问题。

这本质上是个代际鸿沟问题，因为他们是为信息交换而生，而非为语义关系表达而生。

为此，本文提出一种面向语义建模的极简表达范式——Contextual Structure Expression（CSE），并基于该范式设计出语义结构表达优先、可嵌入自然语言的Context Mark Language（CML），在上下文的标记、传输、存储、共享、运算方面展现出显著优势，在语义关系表达的标准化与通用化方向，有广阔潜力。

1. 引言（Introduction）

近年来，大型语言模型（LLM）在自然语言理解与生成领域取得了突破性进展。

背景

随着生成式人工智能（Generative AI）和大语言模型（LLM）的广泛应用，自然语言处理技术正在迈入一个更依赖语义结构和上下文推理的阶段。

然而，当前大语言模型（LLM）的研究与应用关注点，仍主要集中于**知识生命周期管理**，包括知识的获取、表达、生成与对齐等环节。在现有体系中，上下文语义通常作为非结构化、临时性的输入存在，尚未形成系统性的建模范式与中介协议，成为制约大模型智能演进的关键短板。

范式问题

语义的表达范式、存储范式，是当前行业最大的问题。

一方面，当前主流AI模型对知识的学习仍高度依赖离线训练过程，模型参数固化、一致性差，难以支撑高频率的知识版本更新需求。知识一旦蒸馏进模型，成为模型内部固化的权重参数，即成为“冻结语义”，不可修改、不可组合、不可解释，外部系统便难以在语境层级上进行灵活管理。这种结构使得知识迭代的代价极高，尤其对于各类规范文档、时效性强的专业知识而言，**更新缓慢**[1]、滞后严重。

尽管当前广泛采用的 **RAG**（Retrieval-Augmented Generation）方案在一定程度上实现了知识的动态注入[2]，但本质上只是将“冻结方式”从模型内部转移到了外部的检索数据库，形式从权重参数改成了向量碎片，问题并未真正解决。

另一方面，现有的语义表达方式，如RDF、**JSON-LD**等[3]，在结构精度上虽具优势，却不够轻量灵活，标注形态与LLM擅长的自然语言的形态相距甚远，在嵌入、传输、存储、多模态共享、组合运算、线性转换方面有较大局限。

并且，这些方式普遍强调静态结构和显式数值权重标记。静态结构本质上通过**预定义**规范实现语义约束，但不同模态之间、不同模型之间，存在表达规范的差异，导致其天然难以实现通用的跨模态、跨模型的对齐能力[4]。而显式数值权重，难以精准表征自然语言对话与任务中存在的实际隐含关系、语义模糊性与动态意图，是一种"伪精确"。

因此，更具前景的方向，是在逻辑架构上实现**语义结构表达与权重量化的解耦**，允许上下文语义结构独立管理，甚至前移到知识源，在存储阶段保留原始上下文语义，并结合推理时的实际语境（或者不同的语境算法），进行动态组合与权重计算。这意味着，语义结构可以单独演化[5]，而不依赖特定模型或特定模态的权重策略，从而构建可插拔、可演化、可复用、可解释的语义建模体系。

新型语义表达范式（CSE）

为适应LLM时代对语义结构建模的全新需求，CML（Context Mark Language） 提出一种全新的上下文语义表达模型 —— **Contextual Structure Expression**（CSE）。

CSE 采用"语义Token + 关系分隔符"的线性组合方式，仅以一串自然语言兼容的字符串，来承载多维语义信息及其权重结构。这种基于**单字符串**、只**编码语义结构**、**可组合运算**的轻量特征，相比传统静态结构化的标注语言，优势明显：

- 易于**存储**和**嵌入**，能与现有的各种框架、格式无缝集成。
- 与量化无关，不依赖特定模型或模态协议，可作为统一的语义中介格式在多智能体、多模型、多模态之间跨域**共享**语境信息。
- 人类可读，具备"**所见即语义**"的特性，天然具备可解释性，支持语义还原和重构、过程追踪和人机协同。
- **可组合运算**，使其在推理链、动态任务拆解中具备自由拼接与语义调度能力。

这些优势，让CSE有望成为AI时代的"**语义中间语言（Context Interlingua）**"，在未来演化中发挥基础设施作用。

本文内容结构

本文的主要内容结构如下：

- 阐述静态结构语言的语义损耗和代际鸿沟
- 阐述语义冻结的原因是逻辑架构的耦合
- 提出Contextual Structure Expression设计方案（CSE）；
- 介绍Context Mark Language规范（CML）；
- 对比CML和主流表达格式；
- 标准化目标展望；
- 本文涉及或推荐的参考文献。

2. 相关方案与局限（Related Programs and Limitations）

当前，LLM生态中，静态结构语言（如 JSON、YAML、RDF）在多个关键阶段仍发挥着重要作用：

- **数据接入**[6]：用于表达用户画像、知识图谱、API 配置等结构化知识，借助 prompt 模板嵌入模型输入；
- **训练数据构建**[7]：作为监督语料的生成与结构化语义对齐机制（如语义三元组）；
- **推理与任务执行**[8]：将结构化数据重新序列化为自然语言 prompt，服务于 RAG、多模态问答等复杂任务。

但这种价值，更多属于工程价值，而非语义业务价值。

表达范式问题

语义表达层问题。

在LLM产业关注点集中在知识生命周期阶段时（如何接入、训练、推理、更新），JSON、YAML、RDF这类结构语言的确能够满足工程体系的需求，但如果将关注点转向语义生命周期（如何标记、传输、存储、管理、动态运算、语义推理和跨模态对齐[9]），这种工程驱动的方案，局限非常明显。

维度	特征	影响
静态结构	依赖预设 schema 与嵌套结构，语义绑定受限于既定的字段命名与层级结构。	无法提供跨平台一致性，也难以适应 LLM 的动态任务建模。
语义损耗	缺乏时间顺序、意图层级、逻辑递进等语义表达机制，不利于LLM对齐，尤其在复杂场景下更易出现理解偏差。	将结构化语义线性化为prompt，再嵌入规范结构的过程，本质上是“降级式”的桥接，必然导致语义压缩，带来上下文割裂。

静态结构语言的设计初衷在于**信息交换与存储**，而非复杂的**语义建模与动态推理**。

LLM 虽然“看得懂”JSON、YAML，但他们的形态与自然语言相距甚远，`自然语言⇋不同规范结构⇋自然语言`互转过程中的**语义损耗**不可避免，不是也无法成为最佳的语义表达方式。

这实际上是代际鸿沟问题——产业真正需要的是“语义接口”，而不仅仅是“数据接口”。

语义冻结问题

语义存储层问题。

正如引言中提到了，LLM高度依赖离线训练的模式，造成了语义以权重参数的形式被冻结在模型内部，外部系统难以在语境层级上进行低成本的灵活管理，知识更新几乎等于**重训**模型，代价高昂，最终传导到用户侧的现象是知识更新缓慢。

- 一方面，由于权重参数高度绑定模型结构（如层数、注意力机制），随着训练过程中原始语义结构被不可逆压缩，被冻结成特定模型结构的表达，语义**难以复用或跨平台移植**。
- 另一方面，这部分被“冻结”的语义，其表征质量，高度依赖于训练阶段的算法、语料水平，以及知识原文的质量。但因为原始语义结构丢失，运行时无法直接利用新版本的算法、语料或语义标注进行低成本的语义修正。即使知识源主动提供更高质量的语义标注，模型也难以动态吸收。

这并非单纯的工程实现问题，而是源于Transformer架构在语义表示与存储机制上的结构性局限。因此，更具前景的方向，是在逻辑架构上实现**语义结构表达与权重量化的解耦**。

- 将**语义结构**作为一种通用的中间表示，独立于权重参数，外化到模型之外。
- 这种中间表示类似**ONNX**的神经网络通用交换格式的价值定位[10]，但非计算图，而是专注于语义关系表示，用于显式存储原始上下文语义。

解耦后：

- 语义从模型架构中抽离，成为一种跨平台的“格式数据”，可以被不同模型结构、模态系统，统一解析和处理。
- 语义也独立于知识原文，可以单独演化，外部系统也能够直接读取、编辑或扩展，“语义作为服务”（Semantic-as-a-Service）将成为可能。
- 权重参数将不再是知识的唯一载体。当模型的训练和推理可以围绕通用语义结构进行，这就可能催生出一种新型架构的LLM，只负责“表达解释”语义，而不再承担繁重的“知识保存”职责。从知识角度来看，能大幅度降低知识更新成本、提高复用性和灵活性；从 AI工程角度来看，这也意味着更强的可控性、**可解释性**[11] [19]、算法灵活性。

方向

综上，AI时代原生的“**语义中间语言（Context Interlingua）**”，极可能在未来演化中发挥基础设施作用。这是JSON、YAML、RDF之类的传统静态结构语言，所承载不了的目标。

接下来，将介绍本文提出的动态语义建模范式CSE，以及具体标记语言实现CML，并最终展望其在大模型时代语义表达标准化与通用化方向的广阔潜力。

3. CSE表达范式设计

Contextual Structure Expression（CSE）是一种以“上下文维度”为核心，在语义架构、表达形式、语义结构方面，采取了完全不同的范式。

先看样例

表达	语义解释
`message:urgent+alert@user123`	一条消息具有"紧急"+"提醒"性质，面向 user123
`task.create:document@project.alpha`	表示在项目 alpha 中发起创建文档的任务
`question.ai.model+language`	一个关于 AI、模型和语言的复合问题
`error.timeout.network@device`	一个网络错误，类型为 timeout，发生在设备上

以上是CML定义的典型明文语义格式，接近自然语言，可读性一目了然，且在解析重构、推理路径追踪、语义索引、可解释性方面都非常友好！

从CML样例可以看出，CSE的设计目标之一，是能极简、灵活、直观的表达语义化的上下文结构。

线性结构

CSE采用与自然语言兼容，以**单字符串**的线性文本形式，来承载语义结构表达，这种选择的核心考量，是避免`自然语言⇆结构⇆自然语言`互转所造成的**语义损耗**。

基元架构

CSE范式的语义字符串，由一系列`语义Token`+`关系分割符`两种类型的`语义基元`构成。

- **语义Token**，是描述上下文语义的基本单位，可以是任意字词、短句形式的文本片段，对应LLM领域的Token概念，但不是词法性的，而是某个承载了自然语义的抽象维度。
- **关系分隔符**，用于表达这些语义Token之间的语义关系，隐式声明加权分配的优先级。这些符号（`@`、`.`、`:`、`+`、`空格`）是静态约定的，有明确的主次关系和结构语义。

语义表达规则

我们用 `<semantic_token>` 来表示语义Token，`<separator>` 来表示关系分隔符。那么，表达规则这样定义：

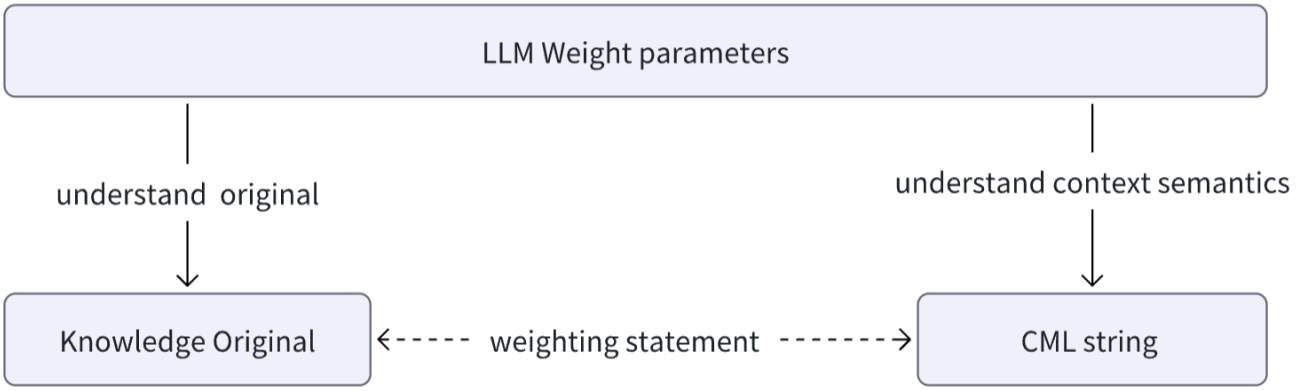
1. 每个 **语义Token**（`<semantic_token>`）必须通过有且只有一个**关系分隔符**（`<separator>`）来相互连接。
2. **语义Token**和**关系分隔符**总是交替出现，`<semantic_token><separator><semantic_token><separator>.....<semantic_token>`。

模糊语义

CSE范式，并不关心显式的数值权重，而是只负责表达原始语义结构，通过Token本身和结构本身的语义，来向LLM声明隐式声明加权项和加权优先级，最终间接影响权重计算。

- 每一个Token，表达了多种可能的语义
- 每一个关系分隔符，定义有一个或多个明确的关系语义。
- 一个关系分隔符允许支持表征多种关系语义，如果不能直接确定要表达的加权优先级，那么借助实际的语义Token，LLM也可以间接推理出实际的语义关系和加权优先级。

这种模糊容忍度，恰恰具有**动态语境适配性**[12]。意味着，上下文语义可以在不同的阶段单独存储、分布式存储，并结合推理时的实际语境（或者不同的语境算法），进行动态组合与权重计算。



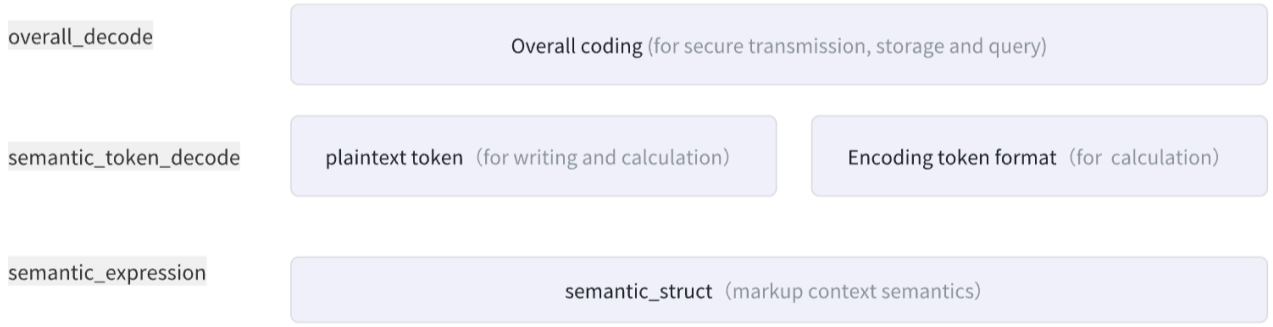
从知识治理架构来看，CSE范式主张的是**知识原文**、**上下文标记**、**LLM权重**数据的"三权分立"，而非把知识原文、语义全部蒸馏到LLM权重中。

格式编码

在自由组合的语义表达基础上，再进行形式化的编码，就构成了最终的语义标记。

- 明文格式讲究人类可读，与Markdown语法兼容，易于编写。

- 同时支持编码格式来消除分割歧义和特殊字符，**免除转义、格式错乱的困扰**[13]。。由于字符串是所有主流格式的基元值，因此能与现有的一切框架、格式、语法无缝集成，只要简单编码，就能被安全用于任何嵌入、嵌套、存储、传输场景，包括但不限于JSON、YAML、HTML、SQL、URL、正则、shell命令……



语义结构

关系分隔符（`<separator>`）在表达形式上负责连接语义Token，同时表达语义结构本身的语义。

具体的关系分隔符是CSE范式设计的核心难点，采用什么样的基元结构，表达什么样的关系语义，采用什么样的运算优先级，这里涉及了一个“**以关系为中心的语义表达**”[14]的最小完备性猜想难题。

这部分，给标记语言留下了设计空间，CML未必是最优解。

4. CML语言规范

CML（Context Mark Language）作为CSE范式的具体实现，在实现上述语义架构、表达形式的基础上，更具体的明确了语义结构的范围、优先级、符号表示，以及语义字符串的编码规则。

符号表示

在形式上，CML选择了`:`、`.`、`@`、`+`、`空格`5个贴近自然语义的**单字节符号**，来作为关系分割符号，并定义了结构语义和明确的运算优先级。

语义结构

5个关系分隔符，是CML围绕Token的**优先级表达**、**结构可扩展**、**逻辑可正交**三个核心需求维度，抽象出的具有高度表达力与组合力的语义结构基元，作为构建一切逻辑语义关系的基础单位。

这5种结构基元，分`基本结构`和`复合结构`、`组合结构`三大类，基本结构描述多个基元语义Token之间的简单结构关系，同时可作为复合结构的组成单元，而组合结构是更宏观的容器类型，可以连接任意语义Token、基本结构和复合结构。

a、补充关系

基本结构。右边语义对象对左边语义对象的补充说明或解释、限制，通常是语义附加，不改变原语义结构

$$f:A\ni B\ni C$$

以符号@来表达，越靠左优先级越高，表示权重应该更高。

比如

- `name`@`identity`@`organization`

- `name`@`company`@`position`

两组标记的权重优先级声明是不同的，都重点都是先强调姓名。

b、线性递进关系

基本结构。一种有序的、逐步推进的语义关系，可以表达各种带方向的轨迹：递进、流向、变换链、指向链、顺序、因果、类型细化、生命周期.....

$$f : A \to B \to C$$

用符号`.`表达，左侧权重是否应该略高于右侧，需要LLM可以结合实际语义token来最终判断权重优先级。

比如

- `生物`.`动物`.`人`，重点落在`人`这个子类上。
- `产品设计`.`开发`.`运营`，结合语义token本身，可以很清楚的判定他描述了一个生命周期语义。

c、并列集合关系

基本结构。多个语义并列，类似集合、对象属性集、多分支描述，无优先级或顺序，可以互调位置。

$$f : \{A, B, C\}$$

用符号`+`表达，权重不分主次先后。

比如

- `男人`+`女人`，相对`人`这个概念，正反组合都一样。
- `姓名`+`年龄`+`username`，都是某个账号的注册信息。

d、映射关系

一种语义结构到另一种语义结构的复合对照，k-v形式，两边都可以使用上述基本关系组合，用于支持二维语义表达

$$f(A, B) \mapsto f(C, D, E)$$

用符号`:`表示。类似于键值对，但构造更自由，无论是key还是value部分，都可以使用基本结构，而不仅仅是基元语义token。

```
<key-context-struct>:<value-context-struct>
```

比如下面都是合法的映射语义结构

- `网站`:`doc-war.com`
- `网站`@`doc-war.com`:`文档战场`@`贡献判断力价值`
- `AI`+`LLM`:`ChatGPT`+`Claude`@`v3.7`
- `ask`.`answer`:`请介绍CML语言?`.`CML语言是符合自然语义的语义结构语言`

特别约束

CML**不支持嵌套映射**，避免带来语义结构本身的解析复杂度。比如`用户:张三:删除+查询`之类的表达，在格式上是非法的。

e、组合关系

多个语义结构组合形成新的语义整体，而不损失其原语义，反之拆分亦然，本质上是一种可运算的“关系结构容器”。

$$f(A) + f(B) = f(A + B)$$
$$f(A) = f(A + B) - f(B)$$

用`空格`表示对任意两个结构的语义叠加。

由于`空格`的语义优先级最低，且左右顺序无优先级影响，他同时也是CML字符串的整体运算符，对明文格式的两个CML字符串使用空格自然拼接成一个新的CML字符串，仍然是一个合法的CML字符串，不影响原语义表达。

$$plaintext(A) + space + plaintext(B) = plaintext(A + space + B)$$

这种可自由拆分—>还原的**无损还原**特征，让CML字符串具备语义运算特征，而不仅仅是语义表达，为标记工作分工协同提供了坚实基础。

特别约束

由于`空格`承担了语义上的无损运算职责，比如`如果 用户:张三怎么样`之类的用法，虽然在格式上是合法的，但一定会破坏无损组合原则，当多个语义字符串被拆分重组之后，会因为位置不同而颠覆原有语义。

运算优先级

关系运算类似于编程语言的表达式解析：从左往右进行词法扫描，然后根据关系分割符的优先级，决定语义运算顺序。

CML定义了明确的优先级，以确保在标记、推理阶段对语义解释的一致性。

$$\text{补充关系} > \text{线性递进关系} > \text{并列集合关系} > \text{映射关系} > \text{组合关系}$$

最小完备性

此五种关系分隔符，源自对自然语言语义表达结构的抽象提炼，亦是对表达歧义与结构可控性问题的深入思考。

自然语言参考

CML定义的5种关系分隔符，参考的是自然语言中隐含的最基本的语义结构（**修饰、顺承、并列、对照、组合**）

```
自然语言示例：
例子：“那个`戴着帽子`C 的`高个子`B `男人`A”          (A←B←C)
例子：“他`起床`A，`穿衣服`B，`出门`C”                (A→B→C)
例子：“我喜欢`苹果`A，`香蕉`B，`橙子`C}。”           {A，B，C}
例子：“`ChatGPT`A和`Claude`B、`DeepSeek`C 都是 `AI`D，也叫`LLM`E”          (A+B+C ↦ D+E)
例子：“`这朵花`A + `很美`B = `这朵花很美`A+B                                (A+B=AB)
```

理论上，借助这5种基元关系的组合，能表达绝大部分逻辑关系，具有相当的完备性。

Token内部语义

对于排除、量词区间、映射嵌套等逻辑关系，CML不做原生支持，下沉至Token 层，来配合**补充关系**进行灵活表达，避免结构污染，带来优先级运算和人类直观阅读的复杂度。

因为Token本身也可以表达结构。

```
自然语言例子："`年龄`大约`18~25岁`，一定`不是中国人`。"          （Token本身可以隐含区间、排除结构）
标记：
❌`age`:`range`:`18-25`          （非法嵌套且毫无必要，完全可以往前或往后合并）
❌`age`:`>18`+`<25`              （毫无必要的嵌套，完全没必要拆这么细）
✅ `age`:`range:18-25`           （在Token内部嵌套是符合自然语义的）
✅ `age`:`18-25`
```

这体现了结构的**极简原则**：

CSE范式并不试图预先显式解决一切结构语义上的歧义，而是在实际表达时，借助关系分隔符所连接的语义Token，作为关系语义的上下文，让LLM推理出最合理的语义关系和对应的权重优先级。

显式结构的核心价值

善用Token内部的语义结构表达，也反映了CML在显式编码上的核心价值。

CML之所以接近于自然语言，而不等于纯自然语言，除了用于加权主干语义，有其表达上的必要性。相比自然语言推理在token切分上的完全不可控，显式分割的本质，是进行语义纵向层级和横向关系的拆分，进而消除关系歧义，最终提升可控性与可解释性。

编码规则

在语义表达的基础上，定义了2种标准的CML字符串模式。核心区别是，在什么阶段，用什么形式，来标记语义Token（semantic_token）本身。

Semantic Original Text	Markdown Format	UTF-8 + Base58 Encoding
token	`` `token` ``	E8uqz5b

自然语言格式

自然语言格式，面向文档工程师的`明文编写`体验，适用于人类可读场景。

以**markdown语法**[15]中的反引号标记（inline code），来包裹语义Token（semantic_token）。文档工程师，可以使用所见即所得的markdown编辑器，作为语义结构的明文编辑环境，可以非常快捷、直观。

比如，用markdown编写明文字符串：

```
`token1`.`token2`@`token3`+`token4` `token5`:`token6`
```

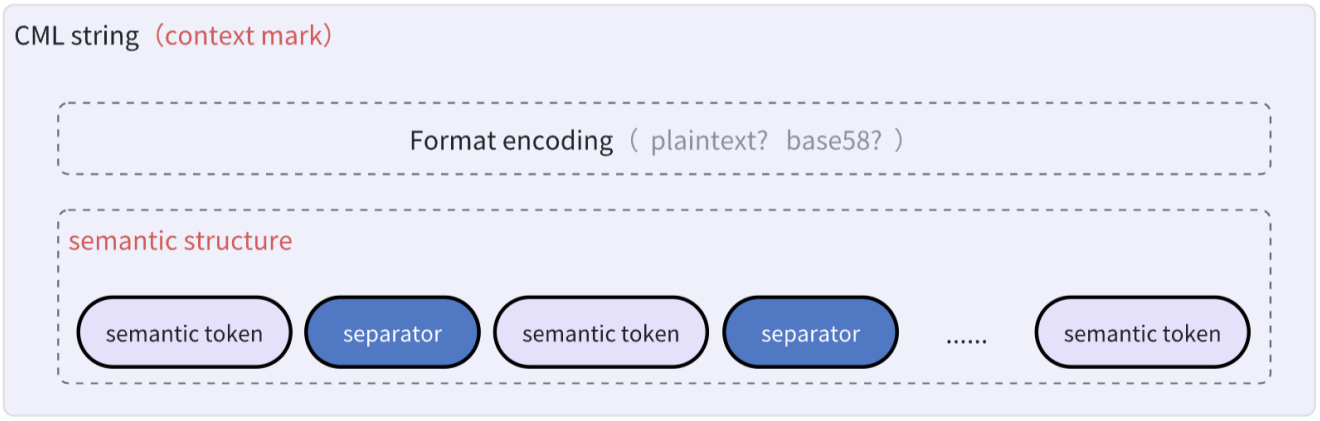
可实时渲染成下面的自然语义效果，一目了然：

`token1`.`token2`@`token3`+`token4` `token5`:`token6`

编码格式

带反引号`的CML字符串，包括分割符号，在某些特殊场景下，可能会带来偏离预期的解析边界和转义要求。

因此，面向`嵌入`、`存储`、`解析`、`运算`场景，CML定义了更安全一致的编码输出格式。



示例

```
`token1`.`token2`@`token3`+`token4` `token5`:`token6`
```

将以上明文CML字符串按如下顺序编码

1. 从CML字符串原文中顺序提取`语义Token`和`关系分隔符`
2. 先将每一个语义Token原文（不包括反引号的token字符串），使用UTF-8编码成字节流，再对字节流进行**Base58编码**[16]，生成Base58字符串

```
token1 → 编码 → zyvFCwFv
token2 → 编码 → zyvFCwFw
token3 → 编码 → zyvFCwFx
token4 → 编码 → zyvFCwFy
token5 → 编码 → zyvFCwFz
token6 → 编码 → zyvFCwG1
```

3. 再用关系分隔符原文重新拼接（这个过程的本质是用base58代替反引号的包裹作用）

```
zyvFCwFv.zyvFCwFw@zyvFCwFx+zyvFCwFy zyvFCwFz:zyvFCwG1
```

4. 最后，再次用UTF-8+Base58进行整体编码，消除一切特殊字符

```
3EkzyE8r5SqnU6KSbLS98LVLJxFoNvskzaazkuEEryWminqaGwJz13YoatvfoRWoDyrofwUCQ
```

换行和多空格问题

在自然编辑中，尤其是长上下文标记场景下，从可读性角度，人类可能倾向于对长字符串进行换行分割，而不是坚持使用空格来分割。因此，CML编辑器应该支持符号等价兼容，在明文格式的解析和存储时，将`\n`、`\r\n`、`\r`等各种可能的换行符号，自动改成`空格`符号。

同理，人类编写场景中，在同一行中，可能希望连续敲击多个空格，以提升结构拆分的直观性，CML编辑器也应该支持等价兼容。

在进行base58编码前，应该明确消除`换行符`和多余的`空格`，统一换成`单空格`符号。

5. CML与其他方案的比较

目前在相关领域尚未发现针对该问题的系统性方案，截止论文发表之日，没有查到任何基于**动态语义表达**的范式论文，或者**上下文标记语言**相关的论文，以及面向**语义生命周期**维度的LLM治理架构设计，或者相关的工具或文档。

行业仍然是以JSON、YAML、RDF为代表的静态结构语言，或者相关的改良方案，来支撑语义表达和存储、交换。

JSON-LD vs CML

其中兼容JSON的JSON-LD算是目前最先进的静态语义表达方式之一，尤其是在 web 语义化和结构化数据标注方面。

一方面，他结合**schema.org**标准词汇表[17]，来表达语义，是主流搜索引擎支持的语义标注方式。另一方面，使用`<script type="application/ld+json">`，可以只向 HTML 页面中 **嵌入一段“机器用”的结构化数据**，它不会被浏览器渲染，也不会影响网页的显示效果或样式布局，提供了一种非侵入式的语义表达能力。

以下是一个典型的JSON-LD格式

```
{
  "@context": "https://schema.org",
  "@type": "Article",
  "headline": "A New Paradigm: Redefining Knowledge Discovery",
  "author": {
    "@type": "Person",
    "name": "lilei"
  }
}
```

但注意，schema.org枚举了超过2000个词汇量，如此庞大的静态语义规范，对人工标注提出了过高的要求，反而不利于标注质量，即使是专业工程师也难以在短时间内熟练掌握其每个词汇及语义细节，在 LLM 时代显得“过重”。

这也是为什么CML只至简定义了5个关系分隔符的原因，我希望CML成为语义标记领域的Markdown，而非HMTL。

JSON Lines vs CML string

并且JSON-LD的嵌套结构非常复杂，在大规模标注场景下解析效率很低，并不是LLM领域的主流。相反，LLM更愿意采用反向的方案**JSON Lines**[18]，来实现 **增量写入/流式处理**（line by line），通过在每个 单行JSON 对象后面紧跟着换行符（`\n`）来分割，来确保每一行可以单独被读取和解析，不用一次性加载整个文件。JSON Lines被 OpenAI、Hugging Face、FastChat 等广泛用作微调、数据训练格式。

以下是一个example.jsonl文件

```
{"id": 1, "prompt": "你好", "response": "你好，请问有什么可以帮助的？"}\n
{"id": 2, "prompt": "什么是JSONL？", "response": "JSON Lines 是一种结构化文本格式。"}\n
{"id": 3, "prompt": "谢谢", "response": "不客气！"}\n
```

相比JSON，JSON Lines的核心改进是借助了`\n`这个组合关系分割符。该思想，与CML采用`空格`作为组合运算符有异曲同工之妙，CML字符串显然天然支持增量写入/流式处理。

但`.jsonl`需要文件存储，不适合用作嵌入式数据的格式，并且，因为每行需要解析为 JSON 对象，因此分割过程涉及更高的计算开销，尤其在数据量很大的时候，相比之下，CML的单字符串切割相当高效。

6. 结论与展望（Conclusion and Future Work）

CSE范式，以及CML的设计，不仅是一种单字符串表达结构的形式革新，更是一种语义表达方法论的思想变革，以及对下一代LLM逻辑架构和语义生命周期的设计机会思考。

展望未来，CSE和CML具备广泛的理论研究价值与工程实践前景，可能成为AI时代的语义基础设施，作为整个生态的结构化语义中间语言而存在。随着大模型能力持续提升，语义表达、组织、推理，将极可能成为下一代 LLM 的核心竞争力与分水岭。

本文建议产业界积极关注本范式的理论框架与潜在变革机会，有更多有识之士参与进来，共同推动 CSE或CML 成为全球通用的语义表达与协同标准。

本文作为笔者个人投身开源领域的首篇系统性研究论文，虽力求严谨和系统性，但作为一名独立研究者，受制于资源、影响力、跨学科协作方面的局限，必有不成熟之处。本文抛砖引玉，希望引发更广泛的学术探讨与工程实践，期待在未来与更多科研团队、产业机构共同完善这一语义范式体系，参与工程实践，推动结构化语义的全面演进。

开源项目地址：https://github.com/ContextMark/CML

参考文献

[1] B. Thompson, “AI will drive the cost of intelligence to zero? Not so fast,” *Medium*, Nov. 10, 2024. [Online]. Available: https://medium.com/the-generator/ai-will-drive-the-cost-of-intelligence-to-zero-not-so-fast-d90d901baf10. [Accessed: Apr. 2025].

[2] S. Pandey, "Retrieval-Augmented Generation (RAG): Improving AI's Answer with External Knowledge," *Medium*, Oct. 24, 2024. [Online]. Available: https://medium.com/@santoshpandey987/retrieval-augmented-generation-rag-improving-ais-answer-with-external-knowledge-7961641e9fde. [Accessed: Apr. 2025].

[3] K. Cagle, "JSON-LD rewrites the Semantic Web," *LinkedIn*, Sep. 13, 2017. [Online]. Available: https://www.linkedin.com/pulse/json-ld-rewrites-semantic-web-kurt-cagle. [Accessed: Apr. 2025].

[4] C. Qian, S. Xing, S. Li, Y. Zhao, and Z. Tu, “DecAlign: Hierarchical Cross-Modal Alignment for Decoupled Multimodal Representation Learning,” *arXiv*, Mar. 14, 2025. [Online]. Available: https://arxiv.org/abs/2503.11892. [Accessed: Apr. 20, 2025].

[5] Wikipedia contributors, “DOGMA,” *Wikipedia, The Free Encyclopedia*, Apr. 20, 2025. [Online]. Available: https://en.wikipedia.org/wiki/DOGMA. [Accessed: Apr. 20, 2025].

[6] S. Pimparkhede et al., “DocCGen: Document-based Controlled Code Generation,” *arXiv*, Jun. 2024. [Online]. Available: https://arxiv.org/abs/2406.11925. [Accessed: Apr. 20, 2025].

[7] N. P. Ding et al., “Knowledge Prompt Chaining for Semantic Modeling,” *arXiv*, Jan. 2025. [Online]. Available: https://arxiv.org/abs/2501.08540. [Accessed: Apr. 20, 2025].

[8] X. Tan et al., “Struct-X: Enhancing Large Language Models Reasoning with Structured Data,” *arXiv*, Jul. 2024. [Online]. Available: https://arxiv.org/abs/2407.12522. [Accessed: Apr. 20, 2025].

[9] Y. Liu et al., “Generative AI-driven Semantic Communication Networks,” *arXiv*, Jan. 2024. [Online]. Available: https://arxiv.org/abs/2401.00124. [Accessed: Apr. 20, 2025].

[10] “Open Neural Network Exchange (ONNX),” *ONNX.ai*, [Online]. Available: https://onnx.ai. [Accessed: Apr. 20, 2025].

[11] Anthropic, “Interpretability Dreams,” *Anthropic*, May 24, 2023. [Online]. Available: https://www.anthropic.com/research/interpretability-dreams. [Accessed: Apr. 20, 2025].

[12] Z. Li, Y. Wang, and H. Chen, “Dynamic Skill Adaptation for Large Language Models,” *arXiv*, Dec. 2024. [Online]. Available: https://arxiv.org/abs/2412.19361. [Accessed: Apr. 20, 2025].

[13] “Escape character,” *Wikipedia*, Apr. 2025. [Online]. Available: https://en.wikipedia.org/wiki/Escape_character. [Accessed: Apr. 20, 2025].

[14] S. Jauhar, “A Relation-Centric View of Semantic Representation Learning,” *Ph.D. dissertation*, Carnegie Mellon University, 2017. [Online]. Available: https://www.cs.cmu.edu/~sjauhar/thesis.pdf. [Accessed: Apr. 20, 2025].

[15] “Markdown,” *Wikipedia*, Apr. 2024. [Online]. Available: https://zh.wikipedia.org/wiki/Markdown. [Accessed: Apr. 2025].

[16] “Base58,” *Wikipedia*, Jan. 2025. [Online]. Available: https://zh.wikipedia.org/wiki/Base58. [Accessed: Apr. 2025].

[17] “Schema.org,” *Wikipedia*, Apr. 2025. [Online]. Available: https://en.wikipedia.org/wiki/Schema.org. [Accessed: Apr. 2025].

[18] “JSON Lines,” *Wikipedia*, Mar. 2025. [Online]. Available: https://en.wikipedia.org/wiki/JSON_streaming#JSONL. [Accessed: Apr. 2025].

[19] C. Singh, J. P. Inala, M. Galley, R. Caruana, and J. Gao, “Rethinking Interpretability in the Era of Large Language Models,” *arXiv preprint*, Jan. 30, 2024. [Online]. Available: https://arxiv.org/abs/2402.01761. [Accessed: Apr. 2025].

特别附注

在语义范式思考方面，最前沿的概念框架可能来自于2022年中国北京邮电大学的牛凯及其同事。他们的工作旨在将通信系统的思维方式从传统的香农范式的“句法层”推进到“语义层”，将关注点由“数据压缩”转移到“语义压缩”。在这一基础逻辑上，他们进一步提出了相应的数学理论。

[20] K. Niu and P. Zhang, “A Mathematical Theory of Semantic Communication: Overview,” *arXiv preprint*, Jan. 25, 2024. [Online]. Available: https://arxiv.org/abs/2401.14160. [Accessed: Apr. 2025].

[21] K. Niu, J. Dai, S. Yao, S. Wang, Z. Si, X. Qin, and P. Zhang, “Towards Semantic Communications: A Paradigm Shift,” *arXiv preprint*, Mar. 13, 2022. [Online]. Available: https://arxiv.org/abs/2203.06692. [Accessed: Apr. 2025].

在自然语言相关的元语言结构研究方面，中国北京语言大学崔希亮教授发表于《语言教学与研究》2002年第5期的文章，提供有一些认知模型，但这种模型是逻辑维度，而非底层结构维度。

[22] X. C. Cui, “认知语言学: 研究范围和研究方法,” *语言教学与研究*, no. 5, pp. 1–7, 2002. [Online]. Available: https://fls.blcu.edu.cn/attach/0/1410161054566649318.pdf. [Accessed: Apr. 2025].