# LLM4DSL: A Method for Domain Specific Language Design With Large Language Models

# Abstract

领域特定语言（DSL）因其在特定领域中的表达能力和易用性，在信息系统工程（ISE）中具备重要的作用。然而，手工设计高质量的DSL往往需要丰富的领域知识和语言设计经验，这不仅费时费力，还依赖领域专家的参与。目前，一些研究者给出了基于本体扩展设计DSL的方法以及基于LLM辅助的DSL自动设计方法。但前者需要较完善的本体设计，且仅能处理简单的领域，后者则需要较多的人为干预。为解决该问题，本文提出了LLM4DSL方法。该方法仅需要输入任意领域语料及对应的简单领域本体，即可自动设计出对应语料较高质量的DSL，该DSL经过人工微调即可直接使用，也可以作为人工设计DSL的重要参考。和现有工作相比，我们的方法对本体要求极低，可接受由LLM自动生成的本体，同时可面向任意领域的语料，并实现DSL的自动化设计。经过实验，我们证明LLM4DSL所设计的DSL具备较高的质量和可用性，在能够正确表述绝大多数输入语料的同时可以被现有DSL语句生成工作识别并具备较高的准确率。我们的方法实现了从领域语料到DSL设计的自动化，为人工设计DSL提供了有价值的参考，提高了设计DSL的效率。LLM4DSL相关的代码和实验数据可以从中得到。

# 1 Introduction

领域特定语言（Domain-specific Languages， DSL）是针对特定领域量身定制的语言，与通用编程语言相比，在其应用领域中，它们在表达能力和易用性方面均有显著提高[1]。由于其强表达能力和易用性的优势，DSL正在成为软件工程（SE）领域内日益受欢迎的研究领域，并且成为诸如Web、生成式编程、模型驱动工程（MDE）、面向语言编程、软件工程等软件开发及系统工程方法学的重要组成部分[2] [52]。尽管对于DSL的设计方法存在较多研究，也发展出了许多工具。然而，目前很多急需DSL的领域并没有较高质量的DSL存在，因为设计泛化性强、用户易用、机器可读的DSL，存在着诸多难点[64]：（1）设计DSL需要深入理解特定领域的知识，包括其概念、规则、约束和术语。获取和抽象这些知识需要领域专家的参与。（2）将复杂的领域知识转换为清晰、简洁的语法规则并非易事，需要语言设计专家来进行。（3）设计高质量的DSL需要投入大量的时间和精力，包括领域知识的获取、语法规则的设计。领域专家和语言设计专家需要沟通协作，进一步增加了DSL设计的成本。

AI特别是大语言模型（LLM）技术的发展，为DSL设计者们开发了一条新道路。与先前的技术相比，LLM在自然语言理解、处理及生成方面展现出非常强大的能力[4]。同时，其强大的上下文学习能力使得其可以通过较少的样本学习迅速获取一个领域的知识[3]。这使得LLM具备了以下能力：（1）通过给LLM输入领域知识，其可以快速理解特定领域，一定程度上替代了领域专家的作用。（2）LLM的训练数据中有许多成熟的DSL（如SQL等），因此其具备迁移这些DSL的良好设计到新的DSL上的潜力。（3）相较于请领域专家、语言设计专家来设计DSL，LLM生成文本的经济成本和时间成本都要低很多。因此，我们认为LLM在DSL设计方面具有巨大的潜力。

本体通常指的是对存在的系统性描述或理论，探讨实体、属性、关系等基本概念。在计算机科学和信息科学中，本体则被定义为一个领域内的概念及其关系的形式化表示。如上面所说，设计DSL需要深入理解特定领域的知识，识别出领域内的核心元素及结构，而这些知识正好可以使用领域本体来表示[6]。因此，我们认为，将领域本体作为知识的一部分输入LLM，可以提高LLM生成DSL的质量。

总的来说，设计DSL的核心是需求分析和语言设计。对DSL设计方法的研究也可以视为对“需求表达模式”和“语言设计规则”的研究。Ana Pescador 和 Juan de Lara[63]设计了DSL-maps工具，可以使用可自定义的转换和目录中的元模型设计模式的推荐，自动地将需求转换为初始的元模型设计。Karsai G等人[51]基于他们自己在开发语言方面的经验，以及参考现有的通用编程语言（GPLs）和建模语言的设计指南，提出了一些有用的指导方针。Günther, Sebastian[62]针对设计和构建内部DSL，提出了六条DSL设计原则和21种DSL设计模式。也有许多研究者利用本体来辅助DSL设计，将对DSL的需求用本体来描述，再定义一套合理的规则来转换出DSL。João M. Sousa Fonseca等人[60]设计了Onto2Gra工具，通过核心的OWL2DSL模块，基于规则地根据本体中定义的概念和关系，自动生成DSL语法。G. Kulagin等人[61] 提出使用“多方面本体论”来进行DSL元模型构建。也就是根据多方面本体中的概念和关系来相应地构造DSL元模型中的各种元素。上述这些方法一定程度上代替了语言设计专家在DSL设计中的作用，但仍然需要领域专家的参与，或要求非常完整且正确的本体。而即便该领域已经存在完善的本体，领域对实际DSL能力的要求很可能超出领域本体涉及到的概念。例如：领域本体只包括这个领域有什么，但DSL要表示对领域实体的“操作”。使用这些本体辅助DSL设计的方法仍然需要领域专家花费大量精力在构造和完善领域本体上。

近年来，本体学习被许多研究者所关注。本体学习是指自动地从自然语言中生成本体的方法。对本体学习的研究可以分为两类：（1）使用传统NLP方法如语言学技术[29]、Statistic技术[30]等，在领域知识中提取术语和关系。（2）利用LLM来进行本体自动生成。相比之下，利用LLM生成本体不需要模型训练和相关的数据集，成本更低且更简便。然而，如上面所说，领域对实际DSL能力的要求很可能超出领域本体涉及到的概念，通过这种方法得到的本体不包含对DSL特定的需求，难以直接提供给上述基于本体设计DSL的方法。

本文提出了LLM4DSL方法，该方法利用LLM，基于领域本体及领域的自然语言语料，生成DSL，为人类设计DSL提供参考和帮助。首先，将领域语料提供给LLM得到领域本体。将本体简化和微调后输入LLM，通过zero-shot自引导提示方法，让LLM根据语料和领域本体给出可以参考的具有良好结构的现有DSL。在此基础上，参考上一步骤给出的DSL的结构，结合领域本体和语料生成对应语料的初始DSL。然后为进一步提高DSL的质量，采用基于建议列表的迭代优化。每次迭代仅让LLM针对初始DSL和部分语料提出修改建议，最后经LLM总结这些建议，修改初始DSL得到最终的DSL。整个过程基本无需人为干预，最终得到的DSL可以作为对应语料的DSL直接使用，或作为设计更完善DSL的参考。经过实验，我们所提出的LLM4DSL方法可以生成平均语料覆盖率（即用DSL可以准确表示的语料占全部语料的比例）达到81%的DSL，该DSL经过微调后输入DSL语句生成的SOTA工作，可以使LLM应用该DSL表示对应语料的准确率平均达到77%。同时，通过参考LLM4DSL的结果设计DSL，相比纯手工设计同等质量的DSL平均节约时间xx。

LLM4DSL方法的主要贡献如下：

1. 和传统的基于本体设计DSL的方法相比，LLM4DSL对输入本体的要求很低，降低了构建本体的成本。因为我们只需要通过本体让LLM对该领域有一个基本的理解，本体中缺失的东西LLM可以在给定的领域语料中了解到。这使得直接使用LLM生成本体的结果成为可能，大大减少了构造领域本体花费的时间与精力。
2. 通过实验评估，LLM4DSL方法设计的DSL具备较高的质量。在覆盖率方面，LLM4DSL设计的DSL能够正确表述绝大多数输入语料。在可用性方面，LLM4DSL设计的DSL可以被DSL语句生成的SOTA工作所识别，并在对应DSL语句生成任务中取得较高准确率。
3. LLM4DSL方法实现了从领域语料到DSL设计的自动化，为人工设计DSL提供了有价值的参考，提高了设计DSL的效率。

# 2 Related Works

## 2.1 DSL Design

领域特定语言（Domain-Specific Language，简称DSL）是一种为特定领域设计的编程语言或规范，它专注于解决某一特定领域的问题，而不是广泛的应用场景。DSL的特点是它的高度抽象性和领域相关性，使得它能够以更加直观、易于理解和操作的方式来表达领域概念和业务逻辑。设计DSL是一个系统化的过程，它涉及理解目标领域的需求、创建能够准确表达领域概念的语言结构，以及确保该语言易于使用和理解。

一直以来，DSL的设计方法受许多研究者所关注。从DSL设计本身上讲，各种各样的方法都围绕着两个核心：需求和语法设计。针对DSL设计中的需求分析阶段和从需求过渡到设计阶段，Ana Pescador 和 Juan de Lara[63]提出了DSL-maps方法。DSL-maps是一种受思维导图启发的符号表示方法，用于表示 DSL的需求。这种符号表示方法由一个工具支持，能够帮助自动将需求转化为初始的元模型设计。开发者可以在此元模型基础上进一步完善和实现DSL。Karsai G等人[51] 基于他们在开发DSL方面的经验，以及通用编程语言（GPLs）和建模语言的设计指南，总结并提出了一系列指导原则。Günther, Sebastian[62] 针对设计和构建内部DSL，提出了六条DSL设计原则和21种DSL设计模式。在各种领域中，具体的DSL设计或多或少地参考或遵循这些设计原则，如Web[52]、分布式系统[54]、工业[55]、IOT[56]等领域中的一些DSL。然而，按照这样的方法设计DSL需要领域专家和语言设计专家大量的参与，花费的时间和精力都很多。

借助领域本体设计DSL也受到许多研究者的关注。Chhaya等研究者[10]指出，提取领域知识、构建领域本体是设计DSL的重要方法。João M. Sousa Fonseca等人[60]设计了Onto2Gra工具，通过核心的OWL2DSL模块，基于规则地根据本体中定义的概念和关系，自动生成DSL语法。Lyadova L N等人[6]和 G. Kulagin等人[61]给出了设计DSL的一种一般方法——多方面本体论。该方法提出两个本体：语言本体和领域本体。其中语言本体ERDLSL作为DSL本体，由三个核心部分组成——实体、关系、属性。之后将对应的领域本体中的实体、关系、属性等与RERDLSL进行映射，即可得到对应领域的DSL。这些借助本体设计DSL的方法，相当于将本体作为DSL设计的需求分析结果。在这种形式化的结果的基础上，设计规则来自动地生成DSL。然而，这种方法十分依赖一个正确且完善的本体。即便该领域已经存在完善的本体，领域对实际DSL能力的要求很可能超出领域本体涉及到的概念。例如：领域本体只包括这个领域有什么，但DSL要表示对领域实体的“操作”。使用这些本体辅助DSL设计的方法仍然需要领域专家花费大量精力在构造和完善领域本体上。

## 2.2 Ontology

本体一词源自哲学，指的是对存在本质的系统性描述。在计算机科学和信息科学中，本体被定义为一个特定领域的概念模型，这些概念及其相互关系被明确地形式化表示。具体来说，本体包括一组术语、属性和关系，用于描述该领域的实体及其特征。通过这种方式，本体可以提供一种结构化的知识表示方法，支持数据的共享、交换和重用。[58]

本体在DSL设计中扮演着至关重要的角色。它能够提供一个结构化的知识表示框架，使得DSL的设计更加系统化和模块化。通过定义领域内的概念、属性及其关系，本体有助于确保DSL能够准确地反映特定领域的复杂性和细节[59]。

Uschold等研究者在1995年首次提出了本体开发方法论的必要性[15][16]。此后，一些方法论和工具被迅速提出并在不同领域得到应用[17,18,19,20,21,22]。这些方法为后续的本体构建方法研究奠定了基础。[34]指出了Competency Question对于本体构建的指导作用。[14]对当时存在的主要本体工程方法论进行了分析和比较，指出构建本体是一个复杂而繁琐的任务，并建议本体工程师和研究者探索新方法与技术以实现更高效的本体设计。

近年来，本体工程取得了很多进展，包括多种本体开发方法的提出、开发过程的系统化以及与敏捷开发方法的结合[23]。[24]提出了本体开发的敏捷方法，[25]进一步提出通过使用数据示例创建文档化的本体，通过迭代的方式降低本体设计所需的精力。[26]提出了本体构建比较完备的方法论，包括本体的提取、聚类、类间层次及关系的分析、DataProperty及ObjectProperty的提取等。[23]介绍了本体工程的LOT方法，该方法由经验丰富的研究者经过多年总结和发展后提出，在多个项目中均得到成功应用，解决了一些本体开发中的重要问题[27]。

然而，上述方法的本体构建均需要本体工程师及领域专家的参与及手动构建，这会耗费较多的时间和资源。因此，本体学习，即自动地从自然语言中生成本体的方法，已经成为近年来研究者所关注的内容[28]。一些语言学技术[29]及Statistic技术[30]等传统NLP方法运用于在领域知识中提取术语和关系，[31]指出机器学习也不失为本体学习的一种有效方法。近期，LLM迅速发展并展现出强大能力[3,4]，这些能力和本体学习的需求非常匹配，因此研究者开始关注LLM在本体生成方向的应用。[35]通过与LLM进行"chatting"，利用迭代的方法提取了特定领域的核心知识。Hamed等研究者[32]利用不同的LLM完成了OL中的三个核心任务：term typing, taxonomy discovery,extraction of non-taxonomic relations。文章指出经过微调，LLM可以作为辅助帮助人类实现本体构建。[33]利用基础的zero-shot等prompt方法，通过LLM生成了特定本体Capability Ontology，并在使用few-shot的实验中取得了接近完美的结果。[7]通过多种prompt方法对不同的LLM进行提示，最终利用所提出的CQbyCQ方法在领域知识直接生成OWL本体的任务中取得了令人印象深刻的结果，证明了LLM可以实现高自动化的本体自动构建以辅助本体工程师的工作。

Saeedizade和Blomqvist[7] 研究了大型语言模型 (LLMs) 在本体工程领域的应用，特别是其生成 OWL 本体建议和替代方案的能力。该研究旨在探索 LLMs 是否可以帮助人类建模者更有效地构建本体。他们发现：（1）GPT-4 是唯一能够生成足够质量 OWL 建议的模型。（2）研究评估了多种提示技术，如思维链 (CoT)、思维图 (GoT) 、分解提示、零样本方法以及CQbyCQ。结果表明，不同的提示技术对 LLMs 的输出质量有显著影响。其中，CQbyCQ方法在生成本体的质量上表现最好。（3）将 LLM 生成的本体与人类建模者的作品进行比较，发现 LLM 生成的本体在质量上至少可以与新手建模者相媲美。

本文提出的方法需要使用本体作为输入，为了节省人工设计本体所花费的时间与精力，我们选择使用CQbyCQ方法来生成本体。鉴于我们是将本体提供给大模型作为一种信息，不必保证本体的完全正确，只要其能够提供一个全貌的领域信息即可。我们认为使用CQbyCQ方法得到的本体完全可以满足要求。

## 2.3 LLM & Prompt Engineering

近年来，大模型已成为自然语言处理领域的研究热点之一。大模型是指参数量庞大、具备强大计算能力和表达能力的神经网络模型。这些模型能够处理大规模的语言任务，如机器翻译、文本生成、情感分析等。其中，最著名的大模型之一是GPT（Generative Pre-trained Transformer），它采用了Transformer结构和预训练-微调的方式进行训练，取得了很好的效果。

大模型的优势之一是能够学习到丰富的语义和语法信息，使得其在各种任务上表现出色。通过大规模预训练，模型可以获得大量的语言知识，这有助于提高模型的泛化能力。在预训练阶段，模型通过学习海量的语料库，掌握了词汇、句法、语义等多个层次的信息。在微调阶段，模型根据具体任务的数据进行微调，使其更好地适应具体任务的特点。这使得LLM具备了以下能力：（1）通过给LLM输入领域知识，其可以快速理解特定领域，一定程度上替代了领域专家的作用。（2）LLM的训练数据中有许多成熟的DSL（如SQL等），因此其具备迁移这些良好设计到新的DSL上的潜力。（3）相较于请领域专家、语言设计专家来设计DSL，LLM生成文本的经济成本和时间成本都要低很多。因此，我们认为LLM在DSL设计方面具有巨大的潜力。

Prompt是用户经过设计后提供给LLM的输入。通过Prompt，用户可以与LLM进行有效交互，引导LLM生成答案[36]。合适的Prompt可以有效地提高LLM生成结果的质量[37，38，39，40，42，45]。LLM具备零样本学习能力，而[39]通过为大模型定位一个角色进行role-play prompt可以提高zero-shot情形下LLM生成的结果质量。LLM的上下文学习能力使得给与少量相关知识提示的few-shot prompt取得较好的结果[36]。[37]提出了Chain of Thought(CoT)方法，通过引导LLM思维，LLM能够在多个任务上产生更好的效果。[40]将CoT与Zero-shot相结合，仅通过增加"think step-by-step"提示，即在多个任务上得到更高质量的结果。[42]在[40]的基础上对任务进行逻辑上的分解，通过利用LLM处理多个分解的小任务使其产生更好的结果。 [46]提出了通过整合外部知识可以提高AI的能力，这个思路目前被广泛应用于prompt engineering。这些方法以及它们的组合可以提高很多LLM生成问题的结果质量，为本文的方法设计带来的很大帮助。

## 2.4 LLM for DSL Generation and Design

目前已经存在利用LLM来设计DSL的相关研究。My M. Mosthaf, Andrzej Wąsowski[57]设计了DSL Assistant，一个利用大型语言模型 (LLM)，来辅助领域特定语言 (DSL) 开发的工具。它使用了 OpenAI 的 GPT-4o 来生成 DSL 语法和示例实例，并支持多种交互模式来完善 DSL 设计，包括自动错误修复。他们的主要贡献为：（1）设计并实现DSL Assistant，帮助用户更轻松地开发 DSL 和示例。（2）通过用户实验评估了 DSL Assistant 的易用性，并发现它成功地帮助用户创建和修改 DSL。（3）通过实验评估发现，生成的 DSL 的质量取决于特定的领域和交互模式。然而，这种方法仍然需要人频繁地参与设计环节，最终结果的质量也与人们使用工具时提供的输入相关。

在完成DSL设计之后，更重要的是DSL的应用。即按照生成的DSL语法产生句子的效果如何？我们选择了利用LLM实现DSL生成方面的最新成果[13]。Wang等人探讨了如何利用大型语言模型（LLMs）进行领域特定语言（DSL）的生成。他们提出了“语法提示”方法，该方法通过将外部知识和领域特定约束（BNF语法的形式表达）融入LLMs的上下文学习中，从而提高模型在少量示例下的生成能力。他们的主要贡献为：（1）提出“语法提示”方法： 通过在每个示例中添加一个专门的BNF语法，该语法仅包含生成特定输出所需的规则子集，从而引导LLMs学习领域特定约束。（2）设计约束解码算法： 设计了一种基于Earley解析的解码算法，确保生成结果符合语法约束，同时尽量减少LLM API调用次数。（3）将“语法提示”应用于语义解析（SMCalFlow、Overnight、GeoQuery）、AI规划（PDDL）和分子生成（SMILES）等任务，并取得了显著的性能提升。我们会将生成的DSL经过人工微调，保证其符合这篇文章的要求，然后作为输入提供给“语法提示”方法。观察生成的结果，以验证我们方法生成的DSL的可用性。

# 3 Method

我们研发了一种方法，称为LLM4DSL。该方法利用LLM，通过输入希望设计DSL的任何自然语言语料以及该语料所对应的领域本体，不需要任何其它输入和人为干预，即可自动化地设计该语料所对应的BNF形式的DSL。人类设计者可以在微调该DSL后，直接将其作为领域的DSL使用，也可以将其作为参考，分析语料的核心内容和结构化表达方式，从而设计更完善的DSL。

LLM会在其预训练的过程中遇到许多元语言（可以用于描述其它领域知识或语言）的语法示例，因此其对这些知识有着一定程度的理解。BNF语法是一种常用的描述各种语言文法的标准元语言，以一定的频率出现在LLM的训练语料库中，LLM也对BNF语法有着较强的理解能力[1]。因此，我们采用BNF语法作为LLM4DSL方法生成的DSL语法。一方面，LLM对BNF的较强理解能力可以使其生成较高质量的DSL，也有助于其对初步生成的DSL进行分析和迭代修改。另一方面，BNF作为目前最常用的语法形式之一，为其定制的解析器也比较多，因此BNF形式的DSL也便于后续应用及与其它工作组合。

LLM4DSL方法主要分为两个部分，分别是初始设计阶段（3.1）和迭代优化阶段（3.2）。在初始设计阶段，我们通过设计zero-shot提示方法，使得LLM4DSL可以仅根据语料和对应的领域本体生成具有一定结构、能够基本表述语料内容的BNF形式的DSL。为了提高设计出的DSL的质量，我们会以基于建议列表的迭代优化方式对其进行优化，从而得到最终的DSL。

## 3.1 Initial Design Phase

我们希望LLM4DSL方法可以根据输入的任意领域自然语言语料全自动地实现对应语料的DSL设计，并得到一个质量较高的初始DSL（我们通过实验证实，若初始DSL的质量不足，即使通过迭代优化，最终也难以获得高质量的DSL）。然而，这存在两个难点。首先，一个高质量的DSL需要能覆盖语料的核心要素，特别是语料中包含的实体及这些实体之间的关系。然而，在没有任何先验知识的情况下，LLM无法保证它能够绝对正确地提取出语料中所包含的核心要素（即使目前LLM已经具备较强大的能力）。一旦LLM对语料的理解出现问题，未能成功解析其核心内容，初始DSL的质量就会迅速下滑。此外，一个好的DSL需要一个良好的语法结构。具体来说，高质量的BNF形式的DSL应当将语料核心内容及其关系以若干非终结符的形式抽象出来，并通过组合这些非终结符实现领域知识的表示，而不是简单地复用语料中的内容。例如，对于语料，"what state borders hawaii ."，存在如下两种DSL语法：

|  |
| --- |
| **Syntax a :** |
| query : = query\_type entity relation entity ". "  query\_type := "what "  entity := state | city  state := "state " [state\_name]  state\_name = "hawaii"  relation := "border" |
| **Syntax b :** |
| query : = what\_border\_query  what\_border\_query := "what " "state " "border " "hawaii " |

用语法a表示对应的语料为"what state border state hawaii ."，用语法b表示语料为"what state border hawaii"。即使语法b比语法a在内容上更简洁、更接近原始语料，我们依然认为语法a是一个更好的DSL。因为设计DSL的最终目的不是为了表示当前的语料，而是为了表示语料所代表的领域知识，而a形式的DSL可以表示任何以"state border state"形式的语料，仅需要为其终结符进行扩展，反之b形式的DSL则需要为每个语料均设计一个语法，这显然是不合理的。然而，为了保证我们的方法可以为所有领域设计DSL，我们不能以few-shot的方式为LLM提供一些示例，因为最终最适合语料的DSL形式可能与我们提供的任何示例都不同。因此，如何提示LLM使其能够生成结构良好的DSL也是一个难点。为解决上述问题，我们引入了领域本体（3.1.1），可以辅助LLM迅速理解语料的核心内容。在此基础上，我们采用了自引导学习方法（3.1.2），使得LLM可以通过zero-shot的方法形成结构良好的高质量DSL。初始设计阶段的Prompt设计见Prompt 3.1

### 3.1.1 Corpus Learning with Ontology

前面我们提到，高质量的DSL需要覆盖语料中的全部实体和实体之间的关系，从而实现对语料的全面表达。因此，我们自然地想到了领域本体。一个语料的本体充分包含了语料的所有实体（Classes）以及这些实体之间的关系（Properties），同时隐去了语料中的非关键内容，使得LLM的注意力可以很好地集中到领域的核心内容上。

具体的prompt方法如Prompt 3.1中的1-3所示，仅需要向LLM提供语料对应的本体即可，不需要额外增加复杂的提示。我们尝试过诸如"具体分析领域本体，并将其中的全部类作为非终结符，并确保在语法中体现出本体中的Property"等提示，但这会造成结果质量下滑（得到的DSL会特别类似本体的形式，同时几乎不具备表述语料含义的能力）。我们认为上述提示可能会将LLM的注意力引向分析本体的具体内容，而忽略其原本的任务。

需要指出，引入本体的这个步骤与我们全自动地实现DSL设计的理念并不相悖。目前，通过各种技术自动化地为语料生成本体的方法已经比较成熟。我们使用[7]中所提出的方法，通过GPT自动地生成了对应语料的rdf turtle形式的DSL，并进行了微调修改，保留其核心内容作为Prompt 4.1的输入。由于rdf turtle是一种非常广泛应用的本体形式，预计其出现在了LLM的训练语料中，因此LLM对其具备较强的理解能力，可以在其辅助下更好地理解语料。

### 3.1.2 Self-guided Learning of DSL Structure

前面已经提到，我们需要引导LLM设计结构良好的DSL，但同时不能采用few-shot learning的方式，因为我们希望LLM4DSL方法能够对任意领域的语料进行DSL设计，而few-shot的方法无疑会限制这个任务的结果，因为我们提供其学习的DSL结构可能都不适合作为现有语料的DSL结构。

为解决上述问题，我们采取利用LLM自身知识的方法进行自引导学习。目前，数万亿参数的LLM的训练数据已经非常庞大，我们认为对于任意为其提供的语料，LLM完全有能力根据其已有的知识，从现有的DSL中（例如SQL语言）得到适合表述该语料的结构。我们使用Prompt 4.1中4-12条prompt实现了LLM的自引导学习。具体来说，我们首先提供领域语料，并引导LLM分析其已知的DSL，尤其是具备较强抽象表述特定领域知识能力的DSL。在此基础上，我们令其选择这些DSL中结构上可以较好地表述当前语料的DSL。由于LLM输出长度的限制，一般其会选择1-2个候选DSL。最后，我们引导其生成这些DSL的BNF形式，作为其后续特定语料DSL设计任务的参考。

需要注意的是，我们强调LLM去分析其所参考的DSL的结构，而不是具体内容，否则LLM的注意力会被转移，并使得其设计出和参考的DSL类似甚至雷同的DSL。

Prompt 3.1 初始设计阶段的Prompt

|  |
| --- |
| Human: |
| 1. Below is the ontology of this field, you should understand it: 2. Ontology: 3. {here input the ontology of the corpus} 4. Below is the corpus you will express: 5. corpus: 6. {here input the corpus} 7. Identify DSLs which are known for their ability to abstractly represent domain-specific concepts. You should refer to these DSLs when designing a new DSL for the following corpus. 8. Provide the BNF form of the DSLs you choose to reference, emphasizing the structure over the specific content. 9. (Your answer should be in the following format): 10. [Reference DSL Start] 11. ... 12. [Reference DSL END] |
| LLM: |
| [Reference DSL Start]{here is the referenced DSL LLM found} [Reference DSL END] |
| Human: |
| 1. corpus:{here input the corpus} 2. Using the structural principles from the referenced DSLs, design a new DSL that is capable of abstractly representing the corpus above. 3. {other prompts to constrain the shape of DSL} |
| LLM: |
| {A preliminary design of DSL} |
| Human: |
| 1. {other prompts to fine\_tune the DSL above} |
| LLM: |
| [DSL Start]{the final DSL designed in the initial generation phase} |

## 3.2 Iterative Optimization Phase

在初始设计阶段LLM设计出的DSL的基础上，我们需要对其进行进一步优化，以提高本方法最终设计的DSL质量。利用LLM对已知结果进行优化的方法有很多，由于我们的目标是在没有人为干预的基础上实现整个流程，因此我们采取了迭代优化的方法。具体来说，我们将提供给LLM的语料进行拆解，得到若干语料段，并在每次迭代中引导LLM完善当前的DSL，确保其可以正确表述本轮次迭代中提供给它的语料段。这样，在迭代完所有的语料段后，LLM设计的DSL应当可以准确表示提供给它的语料。

添加

然而，直接进行迭代是不行的，因为LLM对于上下文的记忆存在一定的限制。而在正常的迭代方式中，LLM每个轮次都会生成对于语料的分析、对原始DSL的分析、修改原始DSL得到的新DSL以及用新的DSL表述语料段等。即使严格约束LLM的输出形式，我们也需要每次迭代都为其提供本体、语料段及当前的DSL，因此一旦迭代轮数较多或语料、DSL较复杂，LLM很容易在后续迭代中遗忘其先前的分析，从而使得其设计的DSL不能充分表示语料的内容。

为解决这个问题，我们采取了基于建议列表的迭代优化方式。首先，我们将语料均等拆分为5个语料段，以限制其迭代轮数。对于每个语料段，我们使用Prompt 3.2引导LLM生成对应的suggestion。具体来说，1-3条prompt依然引入本体，4-8条prompt引入初始设计的DSL及当前的语料段，9引导LLM为初始DSL提供suggestion并输出。这样，经过5轮迭代，我们就得到了一个由5个suggestion组成的suggestion\_list。相比于直接进行迭代，suggestion\_list相当于LLM对DSL的改进执行了一个压缩操作，大幅减小了迭代阶段LLM的输出长度，从而解决了遗忘的问题。

最后，将得到的suggestion\_list与全部的语料以及初始设计阶段得到的DSL提供给LLM，通过Prompt 3.3，引导LLM结合suggestion\_list优化初始设计的DSL，得到最终的DSL。该DSL具有较高的质量，可以作为人类设计对应语料DSL的有效参考。

需要注意，Prompt 4.1-4.3以及其之间的数据处理操作在我们的代码中均被整合，用户只需要运行一个方法即可由语料得到其最终对应的DSL。

Prompt 3.2 生成suggestion的prompt

|  |
| --- |
| Human: |
| 1. Below is the ontology of this field, you should understand it: 2. Ontology: 3. {here input the ontology of the corpus} 4. This is the DSL and a corpus from the same field. 5. DSL: 6. {DSL obtained in the initial generation phase} 7. corpus: 8. {The corpus segments presented to it in this iteration} 9. Please analyze the meaning of the corpus and the expressive ability and the structure of given DSL. Provide suggestions for improving and modifying the DSL to make sure the DSL can STRICTLY EXPRESS THE CORPUS ACCORDING TO ITS GRAMMATICAL RULES. 10. (Output in the following format:) 11. [DSL SUGGEST START]...[DSL SUGGEST END] |
| LLM: |
| [DSL SUGGEST START]{}[DSL SUGGEST END] |

Prompt 3.3 生成最终DSL的prompt

|  |
| --- |
| Human |
| 1. This is the DSL. 2. {DSL obtained in the initial generation phase} 3. This is the corpus. 4. {The entire corpus of the initial input} 5. This is the DSL suggestions: 6. {suggestion\_list consisting of suggestions obtained from five rounds of iterations} 7. Please consider the DSL suggestions comprehensively and modify the DSL step by step. Make sure your DSL can STRICTLY EXPRESS THE FOLLOWING CORPUS ACCORDING TO ITS GRAMMATICAL RULES. 8. (Finally output DSL in the following format:) 9. [DSL START]...[DSL END] |
| LLM： |
| [DSL START]{final dsl}[DSL END] |

4 Experimental Evaluation

本段我们将对我们提出的LLM4DSL方法进行实验验证。首先，我们会介绍我们使用的实验数据集及实验设置。之后，我们提出了四个RQ来验证我们的方法：RQ1: LLM4DSL方法能否为不同领域的语料设计较高质量的DSL? RQ2: LLM4DSL方法设计出的DSL可用性如何？RQ3: 领域本体在LLM4DSL中具体发挥了什么作用? RQ4: 基于建议列表的迭代优化与直接迭代相比具有哪些优势? RQ5: 在LLM4DSL方法辅助下设计DSL与人工手工设计DSL相比能否带来效率的提升？

## Experimental Settings

我们使用如下的数据集作为我们的领域语料: Overnight-Blocks，一系列在合成物块的场景中对于物块不同特征和关系的查询。Overnight-Basketball，包含一系列球员在nba不同赛季的得分、表现等相关的查询。GeoQuery，包含一系列基于美国地理数据库的，关于各个州、城市及河流等内容的查询。我们的数据来自目前DSL Generation领域SOTA工作Grammar Prompting[13]的公开代码库。通过实验，我们发现目前OpenAI公司所发布的GPT-4模型能为我们LLM4DSL方法带来最好的效果，因此我们使用GPT-4作为支撑我们方法的LLM。

对于RQ1，我们在三个领域的数据集中分别选取20条语料，作为LLM4DSL方法的输入语料，并设计了对应领域本体。之后，通过LLM4DSL方法设计出的DSL的语料覆盖量（即通过DSL能够准确表述的语料数目）及过拟合数目（语法中连续包含四个和语料中相同的单词的情况的数目）作为所设计的DSL质量的评判标准。对于RQ2，我们将LLM4DSL为各语料设计的DSL进行简单的完善（人工微调及结构和内容，以使其符合Grammar Prompting方法的Parser）。将该DSL及对应语料输入至Grammar Prompting方法，将其中的10个语料及其DSL表述作为训练，另外10个语料及其DSL表述作为测试，以Grammar Prompting自动生成的DSL表示的准确率作为DSL可用性的判断标准。因为该准确率越高，说明机器对LLM4DSL方法所设计的DSL的理解能力越强。同时，作为比较，我们也在使用相同训练和测试语料的基础上测试了数据集自带的DSL。对于RQ3，我们选取Geoquery数据集，在其它内容完全相同的基础上，删除LLM4DSL方法中和本体相关的内容，进行DSL设计，并与原始方法所设计的DSL的语料覆盖量及过拟合数目两个指标进行比较。对于RQ4: 我们将LLM4DSL方法中基于建议列表的迭代优化修改为朴素的迭代优化，并比较最后生成的DSL质量。对于RQ5: 我们邀请学院中具备本体设计及DSL相关知识的同学分为两组，一组在LLM4DSL方法的辅助下设计DSL，另一组完全手工设计DSL，并分别统计两组同学为同一领域设计出能够完全表达语料的DSL所花费的平均时间。

## RQ1: Quality of DSL Designed by LLM4DSL

我们分别在Basketball、Blocks以及Geoquery三个数据集中各选取20条语料，用LLM4DSL方法设计对应语料的DSL，并分别统计其语料覆盖量和过拟合数目。对于每个数据集语料，我们重复进行了5次实验，结果分别由图 5.1、图 5.2和图 5.3所示。其中，LLM4DSL方法为Blocks数据集设计的DSL表现最好，平均可以覆盖17条语料，而对于其他两个数据集，也可以做到平均覆盖15.6条语料。需要注意的是，在本体的引导下，这些DSL基本可以体现语料的全部核心内容，对于没有被覆盖的语料，在现有语法上略微修改即可实现DSL对它们的表述能力。例如，对于语料"number of points over a season of player kobe bryant from season 2004"，LLM4DSL的方法可能可以表示"number of points over a season of player kobe bryant",同时也具备表示season和具体年份的非终结符，因此只要增加一些非终结符组合即可实现正确修改，而这对于熟悉DSL设计的人来说是非常容易的。同时，LLM4DSL所设计的DSL产生的overfitting非常少，对于结果最差的Blocks数据集也仅产生了平均2.2条overfitting，而对Geoquery数据集所设计的DSL则没有产生任何overfitting。这表明了我们的方法所设计的DSL不仅能够正确表达语料的内容，也具备一个比较好的结构，不会因为过拟合而导致这些DSL的可用性降低。

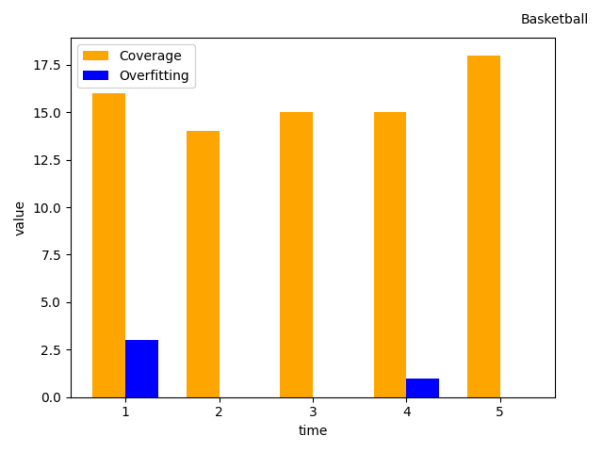


图 5.1 Coverage and Overfitting\_num for Basketball corpus

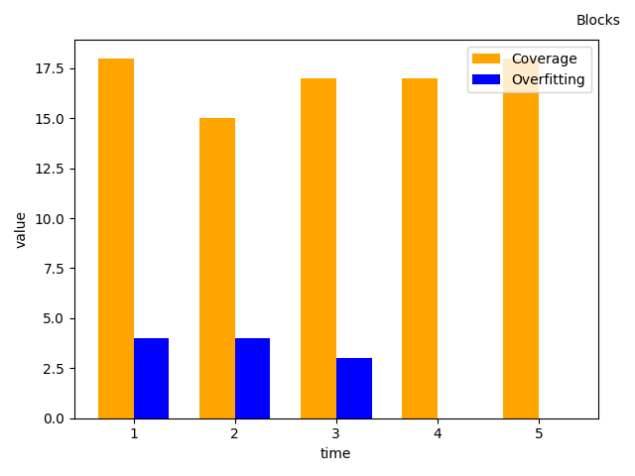


图 5.2 Coverage and Overfitting\_num for Blocks corpus

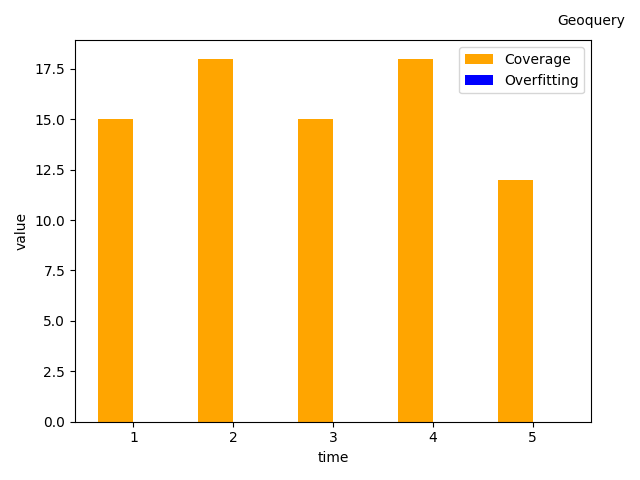


图 5.3 Coverage and Overfitting\_num for Geoquery corpus

## RQ2: Usability of DSL designed by LLM4DSL

我们希望LLM4DSL方法能够辅助人类设计者进行DSL的设计，而且这些DSL应当具有较高的可用性。[13]提出了Grammar Prompting方法，是目前DSL生成领域的SOTA方法，可以用few-shot的方式学习输入的语料及对应的DSL表示，并自动化地生成同领域新语料的DSL表示且具备较高的准确度。因此，我们希望人们在LLM4DSL方法辅助下所设计的DSL可以被Grammar Prompting方法充分识别。这样，在表明LLM4DSL方法的可用性同时，意味着从根据语料设计DSL，到为语料生成其DSL表示的全流程均已在LLM的支撑下实现了自动化。

我们选取了Blocks数据集和Geoquery数据集，均在[13]为其拆分好的训练集和测试集中各选取10条语料，将此20条语料输入LLM4DSL方法，设计其对应的DSL。之后，我们微调了其设计的DSL，使其可以正确表示全部的语料，并通过Grammar Prompting的语法解析器。这个过程非常简单，在保持原始DSL结构的基础上，仅需要简单地为DSL增加部分符号，并删除其冗余内容（LLM为保证DSL的正确性，可能会生成一些类似的语法）。之后，通过一个基于规则的算法，即可将DSL处理为可被Grammar Prompting语法解析器解析的状态。接下来，我们严格遵循上述DSL的语法表达对应的语料，并将原本属于训练集的10条语料及其对应表达作为新的训练集、原本属于测试集的语料及其对应表达作为新的测试集输入Grammar Prompting方法。同时，为了进行比较，我们在保持训练集和测试集语料不变的基础上，使用数据集提供的原始DSL作为对照。对于每个数据集，我们均进行了5次重复实验。

实验结果如图5.4所示，可见对于两个领域的语料，Grammar Prompting方法均可以利用我们在LLM4DSL方法基础上得到的DSL准确地表达语料，准确度分别平均达到0.76/1和0.78/1，均超过数据集自带的DSL。这表明，LLM4DSL方法所设计的DSL具备较高的可用性。我们需要强调，Grammar Prompting方法所测定的语料表示准确度是绝对严格的，和标准表示存在任何出入即视为表达错误。例如在一次实验中，对于语料"number of citizens in boulder ?",其正确的DSL表示应为"what number of citizen\_of boulder .",而Grammar Prompting方法给出的预测为"how\_many number of citizen\_of boulder ."。Grammar Prompting将其统计为错误预测，但实际上只需要在其基础上结合语法进行简单修改，即可得到正确的DSL表示。因此我们可以认为，结合LLM4DSL方法及Grammar Prompting方法，已实现了任意特定领域DSL设计及应用的全流程自动化。

|  |  |
| --- | --- |
|  |  |

图 5.4 使用Grammar Prompting方法分别为两种DSL进行语料表达预测

## RQ3: Characteristics of Ontology in LLM4DSL

为证明本体在LLM4DSL方法中起到的关键作用，我们删除了LLM4DSL方法中和本体相关的提示和输入，并保证其它内容完全不变。在此基础上，我们用裁剪的LLM4DSL方法为Geoquery语料生成DSL，并测定其设计的DSL质量。进行多次实验的结果如表5.1所示。

可见，在平均语料覆盖和过拟合语法数目中，去除本体的LLM4DSL表现均不如原始的LLM4DSL方法。同时，去除本体的方法在实验过程中也产生了3次错误，在这几次实验中，去除本体的LLM4DSL方法直接将原始语料略微修改后作为DSL输出（即完全过拟合）。上述结果表明，本体能够帮助LLM迅速理解语料的核心内容，这将为LLM设计DSL带来很大帮助，使其设计的DSL质量上全方位得到提升。

表 5.1 LLM4DSL去除本体后与原方法效果比较

|  |  |  |  |
| --- | --- | --- | --- |
|  | avg\_coverage | avg\_overfitting | error\_time |
| LLM4DSL  LLM4DSL without Ontology | **15.6**  13 | **0**  1.4 | **0**  3 |

## RQ4: Characteristics of Suggestion List in LLM4DSL

为证明基于建议列表的DSL迭代优化在LLM4DSL方法中的作用，我们将方法中的基于建议列表的迭代方式修改为朴素的迭代方式：每次为LLM提供上一次修改后的DSL与当前的部分语料段，要求LLM在保持上一轮DSL的全部表述能力的基础上进行一次优化，使得新的DSL可以准确地表述本轮次迭代提供给它的语料段。除此之外，方法的其余内容完全不变。在此基础上，我们用修改后的LLM4DSL方法为Geoquery语料设计DSL，结果如表5.2所示。

由实验结果可知，普通的迭代方法在LLM4DSL中效果很差，不仅平均语料覆盖远小于基于建议列表的迭代，也会产生更多的过拟合与错误。这是因为对于LLM来说，每一次迭代实际上都是一个单独的任务，其所依据的内容仅包括本体、上一轮迭代的DSL以及当前提供给它的语料段。由于本体难以反映语料的全部内容，因此LLM在迭代过程中难以把握DSL的整体架构和模式，容易因为attention而陷入局部最优，使得其在每轮次的迭代中倾向于修改DSL使其可以完全准确地表示当前的语料段，而破坏DSL的整体结构和表示能力。而基于建议列表的迭代方法则可以很好地避免这个问题，因为最后提供给LLM的是初始的DSL（结构较好，表示能力较差）和针对每个语料给出的修改建议，这使得LLM可以对DSL的结构和要表示的内容具备一个整体的理解，从而可以设计出高质量的DSL。

表 5.2 LLM4DSL使用基于建议列表的迭代与使用朴素迭代效果比较

|  |  |  |  |
| --- | --- | --- | --- |
|  | avg\_coverage | avg\_overfitting | error\_time |
| Iterate with suggestion list  Formal iterate | **15.6**  5.0 | **0**  3.2 | **0**  4 |

# 5 Discussion

我们的工作存在一些需要讨论的内容。首先，LLM4DSL仍存在一定的生成结果不稳定现象。以Geoquery数据集所进行的实验为例，随机进行的5次实验中，我们得到的平均语料覆盖率是15.6/20，但实际上，我们得到的结果中有两次较好的结果可以达到18/20的覆盖率，而有一次覆盖率为12/20。同时，每次设计的DSL在语法形态上也存在一定差异，在部分实验中生成的结果类似于SQL语法，在其它实验中的语法则更接近自然语言。需要强调，我们已经通过prompt设计尽可能避免了低质量的DSL设计，但LLM的幻觉问题及结果的不确定性实际上是难以绝对避免的，特别是对于LLM4DSL这种多领域的、需要LLM创造及设计能力的任务。因此，在实际使用中，对于同一语料可以进行多次LLM4DSL方法的DSL设计，并综合其结果作为参考。此外，LLM4DSL方法对于输入语料的长度是有限制的，这个限制主要取决于所采用的LLM对token长度的限制。因此，对于模式特别复杂、难以在较少的语料中涵盖其全部模式的领域，需要对较多语料进行拆解，多次使用LLM4DSL方法为每部分语料设计DSL，并将这些DSL进行整合以得到最终结果。LLM4DSL为这些领域所设计的DSL质量可能会低于其为简单领域设计的DSL，后续可能需要一定的人工调整才能得到可用的DSL。当然，LLM4DSL方法的有效性并没有受到威胁，我们的方法可以为一般领域的DSL设计提供很大的帮助。

# Conclusion

本文提出了一种利用 LLM 和领域本体设计 DSL 的方法LLM4DSL，旨在辅助人类设计高质量的 DSL。实验结果表明，该方法能够并生成具有参考价值的 DSL 方案，并有效降低 DSL 设计的成本和时间。其中，本方法提出的将本体提供给LLM，以及在LLM迭代时生成修改建议而非结果，也都通过实验验证了其有效性。另外，我们也通过将LLM4DSL设计的DSL经过人工微调接入“Grammar Prompting”的方法中，进一步验证了LLM生成DSL的可用性。

未来可以进一步开展以下工作：1. 开发 DSL 设计工具，将本文提出的方法工具化，方便用户使用。2. 在其他LLM上进行实验，探索LLM的能力。

# References

1. Marjan Mernik, Jan Heering, and Anthony M. Sloane. 2005. When and how to develop domain-specific languages. ACM Comput. Surv. 37, 4 (December 2005), 316–344. https://doi.org/10.1145/1118890.1118892
2. Kosar T, Bohra S, Mernik M. Domain-specific languages: A systematic mapping study[J]. Information and Software Technology, 2016, 71: 77-91.
3. Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models[J]. arXiv preprint arXiv:2001.08361, 2020.
4. Karanikolas N, Manga E, Samaridi N, et al. Large language models versus natural language understanding and generation[C]//Proceedings of the 27th Pan-Hellenic Conference on Progress in Computing and Informatics. 2023: 278-290.
5. Iung A, Carbonell J, Marchezan L, et al. Systematic mapping study on domain-specific language development tools[J]. Empirical Software Engineering, 2020, 25: 4205-4249.
6. Lyadova L N, Sukhov A O, Nureev M R. An ontology-based approach to the domain specific languages design[C]//2021 IEEE 15th International Conference on Application of Information and Communication Technologies (AICT). IEEE, 2021: 1-6.
7. Saeedizade M J, Blomqvist E. Navigating Ontology Development with Large Language Models[C]//European Semantic Web Conference. Cham: Springer Nature Switzerland, 2024: 143-161.
8. Autili M, Grunske L, Lumpe M, et al. Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar[J]. IEEE Transactions on Software Engineering, 2015, 41(7): 620-638.
9. Ballantyne M, King A, Felleisen M. Macros for domain-specific languages[J]. Proceedings of the ACM on Programming Languages, 2020, 4(OOPSLA): 1-29.
10. Chhaya B, Jafer S. Scenario-based generation of ontologies for domain-specific languages[C]//2020 Spring Simulation Conference (SpringSim). IEEE, 2020: 1-11.
11. Piereder C, Fleck G, Geist V, et al. Using AI-Based Code Completion for Domain-Specific Languages[C]//International Conference on Product-Focused Software Process Improvement. Cham: Springer Nature Switzerland, 2023: 227-242
12. Chen M, Zhang H, Wan C, et al. On the effectiveness of large language models in domain-specific code generation[J]. arXiv preprint arXiv:2312.01639, 2023.
13. Wang B, Wang Z, Wang X, et al. Grammar prompting for domain-specific language generation with large language models[J]. Advances in Neural Information Processing Systems, 2024, 36.
14. Iqbal R, Murad M A A, Mustapha A, et al. An analysis of ontology engineering methodologies: A literature review[J]. Research journal of applied sciences, engineering and technology, 2013, 6(16): 2993-3000.
15. Uschold M, King M. Towards a methodology for building ontologies[M]. Edinburgh: Artificial Intelligence Applications Institute, University of Edinburgh, 1995.
16. Fernández-López M, Gómez-Pérez A. Overview and analysis of methodologies for building ontologies[J]. The knowledge engineering review, 2002, 17(2): 129-156.
17. Fernández-López M, Gómez-Pérez A, Juristo N. Methontology: from ontological art towards ontological engineering[J]. 1997.
18. Gruninger M. Methodology for the design and evaluation of ontologies[C]//Proc. IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing. 1995.
19. Farquhar A, Fikes R, Pratt W, et al. Collaborative ontology construction for information integration[R]. Technical Report KSL-95-63, Stanford University Knowledge Systems Laboratory, 1995.
20. Swartout B, Patil R, Knight K, et al. Toward distributed use of large-scale ontologies[C]//Proc. of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems. 1996, 138(148): 25.
21. Noy N F, McGuinness D L. Ontology development 101: A guide to creating your first ontology[J]. 2001.
22. De Nicola A, Missikoff M, Navigli R. A proposal for a unified process for ontology building: UPON[C]//International conference on database and expert systems applications. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005: 655-664.
23. Poveda-Villalón M, Fernández-Izquierdo A, Fernández-López M, et al. LOT: An industrial oriented ontology engineering framework[J]. Engineering Applications of Artificial Intelligence, 2022, 111: 104755.
24. De Nicola A, Missikoff M. A lightweight methodology for rapid ontology engineering[J]. Communications of the ACM, 2016, 59(3): 79-86.
25. Peroni S. A simplified agile methodology for ontology development[C]//OWL: Experiences and Directions–Reasoner Evaluation: 13th International Workshop, OWLED 2016, and 5th International Workshop, ORE 2016, Bologna, Italy, November 20, 2016, Revised Selected Papers 13. Springer International Publishing, 2017: 55-69.
26. Bravo M, Hoyos Reyes L F, Reyes Ortiz J A. Methodology for ontology design and construction[J]. Contaduría y administración, 2019, 64(4).
27. Tudorache T. Ontology engineering: Current state, challenges, and future directions[J]. Semantic Web, 2020, 11(1): 125-138.
28. Asim M N, Wasim M, Khan M U G, et al. A survey of ontology learning techniques and applications[J]. Database, 2018, 2018: bay101.
29. Jiang X, Tan A H. CRCTOL: A semantic‐based domain ontology learning system[J]. Journal of the American society for information science and technology, 2010, 61(1): 150-168.
30. Rani M, Dhar A K, Vyas O P. Semi-automatic terminology ontology learning based on topic modeling[J]. Engineering Applications of Artificial Intelligence, 2017, 63: 108-125.
31. Petrucci G, Rospocher M, Ghidini C. Expressive ontology learning as neural machine translation[J]. Journal of Web Semantics, 2018, 52: 66-82.
32. Babaei Giglou H, D’Souza J, Auer S. LLMs4OL: Large language models for ontology learning[C]//International Semantic Web Conference. Cham: Springer Nature Switzerland, 2023: 408-427.
33. da Silva L M V, Köcher A, Gehlhoff F, et al. On the Use of Large Language Models to Generate Capability Ontologies[J]. arXiv preprint arXiv:2404.17524, 2024.
34. Ren Y, Parvizi A, Mellish C, et al. Towards competency question-driven ontology authoring[C]//The Semantic Web: Trends and Challenges: 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings 11. Springer International Publishing, 2014: 752-767.
35. Tang Y, Da Costa A A B, Zhang X, et al. Domain knowledge distillation from large language model: An empirical study in the autonomous driving domain[C]//2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2023: 3893-3900.
36. Brown T B. Language models are few-shot learners[J]. arXiv preprint arXiv:2005.14165, 2020.
37. Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models[J]. Advances in neural information processing systems, 2022, 35: 24824-24837.
38. Lu Y, Bartolo M, Moore A, et al. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity[J]. arXiv preprint arXiv:2104.08786, 2021.
39. Kong A, Zhao S, Chen H, et al. Better zero-shot reasoning with role-play prompting[J]. arXiv preprint arXiv:2308.07702, 2023.
40. Kojima T, Gu S S, Reid M, et al. Large language models are zero-shot reasoners[J]. Advances in neural information processing systems, 2022, 35: 22199-22213.
41. Chia Y K, Chen G, Tuan L A, et al. Contrastive chain-of-thought prompting[J]. arXiv preprint arXiv:2311.09277, 2023.
42. Wang L, Xu W, Lan Y, et al. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models[J]. arXiv preprint arXiv:2305.04091, 2023.
43. Long J. Large language model guided tree-of-thought[J]. arXiv preprint arXiv:2305.08291, 2023.
44. Yao S, Yu D, Zhao J, et al. Tree of thoughts: Deliberate problem solving with large language models[J]. Advances in Neural Information Processing Systems, 2024, 36.
45. Hulbert D. Using Tree-of-Thought Prompting to boost ChatGPT’s reasoning[J]. 2023.
46. Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks[J]. Advances in Neural Information Processing Systems, 2020, 33: 9459-9474.
47. Zhang Z, Zhang A, Li M, et al. Automatic chain of thought prompting in large language models[J]. arXiv preprint arXiv:2210.03493, 2022.
48. Wu T, Terry M, Cai C J. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts[C]//Proceedings of the 2022 CHI conference on human factors in computing systems. 2022: 1-22.
49. Yao S, Zhao J, Yu D, et al. React: Synergizing reasoning and acting in language models[J]. arXiv preprint arXiv:2210.03629, 2022.
50. Shinn N, Cassano F, Gopinath A, et al. Reflexion: Language agents with verbal reinforcement learning[J]. Advances in Neural Information Processing Systems, 2024, 36.
51. Karsai G, Krahn H, Pinkernell C, et al. Design guidelines for domain specific languages[J]. arXiv preprint arXiv:1409.2378, 2014.
52. Visser E. WebDSL: A case study in domain-specific language engineering[J]. Generative and Transformational Techniques in Software Engineering II: International Summer School, GTTSE 2007, Braga, Portugal, July 2-7, 2007. Revised Papers, 2008: 291-373.
53. Kuhrmann M. User assistance during domain-specific language design[C]//FlexiTools workshop. 2011.
54. van Amstel M, van den Brand M, Engelen L. An exercise in iterative domain-specific language design[C]//Proceedings of the joint ERCIM workshop on software evolution (EVOL) and international workshop on principles of software evolution (IWPSE). 2010: 48-57.
55. Urquhart, N., Hunter, K. (2024). Evolving Staff Training Schedules Using an Extensible Fitness Function and a Domain Specific Language. In: Smith, S., Correia, J., Cintrano, C. (eds) Applications of Evolutionary Computation. EvoApplications 2024. Lecture Notes in Computer Science, vol 14634. Springer, Cham. https://doi.org/10.1007/978-3-031-56852-7\_6
56. C. G. García, D. Meana-Llorián, V. García-Díaz, A. C. Jiménez and J. P. Anzola, "Midgar: Creation of a Graphic Domain-Specific Language to Generate Smart Objects for Internet of Things Scenarios Using Model-Driven Engineering," in IEEE Access, vol. 8, pp. 141872-141894, 2020, doi: 10.1109/ACCESS.2020.3012503.
57. My M. Mosthaf and Andrzej Wasowski. 2024. From a Natural to a Formal Language with DSL Assistant. In Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24). Association for Computing Machinery, New York, NY, USA, 541–549. <https://doi.org/10.1145/3652620.3687811>
58. Guarino, N., Oberle, D., Staab, S. (2009). What Is an Ontology?. In: Staab, S., Studer, R. (eds) Handbook on Ontologies. International Handbooks on Information Systems. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-540-92673-3_0>
59. Walter, T., Ebert, J. (2009). Combining DSLs and Ontologies Using Metamodel Integration. In: Taha, W.M. (eds) Domain-Specific Languages. DSL 2009. Lecture Notes in Computer Science, vol 5658. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-642-03034-5_8>
60. João M. Sousa Fonseca, Maria João Varanda Pereira, and Pedro Rangel Henriques. Converting Ontologies into DSLs. In 3rd Symposium on Languages, Applications and Technologies. Open Access Series in Informatics (OASIcs), Volume 38, pp. 85-92, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2014) <https://doi.org/10.4230/OASIcs.SLATE.2014.85>
61. G. Kulagin, I. Ermakov and L. Lyadova, "Ontology-Based Development of Domain-Specific Languages via Customizing Base Language," 2022 IEEE 16th International Conference on Application of Information and Communication Technologies (AICT), Washington DC, DC, USA, 2022, pp. 1-6, doi: 10.1109/AICT55583.2022.10013619.
62. Günther, Sebastian. “Design Patterns and Design Principles for Internal Domain-Specific Languages.” (2013).
63. Ana Pescador and Juan de Lara. 2016. DSL-maps: from requirements to design of domain-specific languages. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE '16). Association for Computing Machinery, New York, NY, USA, 438–443. <https://doi.org/10.1145/2970276.2970328>
64. Mernik M, Heering J, Sloane A M. When and how to develop domain-specific languages[J]. ACM computing surveys (CSUR), 2005, 37(4): 316-344.