软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(12):5438−5455 [doi: 10.13328/j.cnki.jos.007409] [CSTR: 32375.14.jos.007409]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于时序逻辑的需求文本隐含语义解析与推理*

李春奕, 马 智, 武 强, 王小兵, 赵 亮

(西安电子科技大学 计算机科学与技术学院, 陕西 西安 710071)

通信作者: 王小兵, E-mail: xbwang@mail.xidian.edu.cn; 赵亮, E-mail: lzhao@xidian.edu.cn

摘 要: 时序逻辑已被广泛应用于形式化验证和机器人控制等领域, 但是对于非专家用户来说难以掌握使用. 因此, 采用自动化手段从自然语言文本中提取时序逻辑公式, 是至关重要的. 然而, 现有工作受限于需求样本稀疏和自然语言语义模糊等因素, 导致其难以准确地识别自然语言文本中隐含的时序语义, 进而造成最终得到的时序逻辑公式错误表达了原始自然语言的语义. 为了解决该问题, 提出一种基于小样本网络的时序逻辑语义分析方法 FSLNets-TLSA, 它采用了数据预处理用来增强文本时序语义逻辑特征, 网络结构由编码器、归纳模块和关系模块组成, 旨在捕捉需求文本的隐含时序逻辑语义信息, 并集成模型增强模块识别监控语义准确度. 在 3 个公开数据集 3 533 个需求样本上与相似工具上完成实验评估, 其分析的平均准确率、召回率和 $F1$ 值达到了 96.55%, 96.29% 和 96.42%, 验证了所提方法的有效性.

关键词: 时序逻辑; 语义分析; 形式化规约

中图法分类号: TP311

中文引用格式: 李春奕, 马智, 武强, 王小兵, 赵亮. 基于时序逻辑的需求文本隐含语义解析与推理. 软件学报, 2025, 36(12): 5438–5455. http://www.jos.org.cn/1000-9825/7409.htm

英文引用格式: Li CY, Ma Z, Wu Q, Wang XB, Zhao L. Implicit Semantic Parsing and Reasoning of Requirement Text Based on Temporal Logic. Ruan Jian Xue Bao/Journal of Software, 2025, 36(12): 5438–5455 (in Chinese). http://www.jos.org.cn/1000-9825/7409.htm

Implicit Semantic Parsing and Reasoning of Requirement Text Based on Temporal Logic

LI Chun-Yi, MA Zhi, WU Qiang, WANG Xiao-Bing, ZHAO Liang

(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: Temporal logic has been extensively applied in domains such as formal verification and robotics control, yet it remains challenging for non-expert users to master. Therefore, the automated extraction of temporal logic formulas from natural language texts is crucial. However, existing efforts are hindered by issues such as sparse sample availability and the ambiguity of natural language semantics, which impede the accurate identification of implicit temporal semantics within natural language texts, thus leading to errors in the translation of the original natural language semantics into temporal logic formulas. To address this issue, a novel method for temporal logic semantic analysis based on a few-shot learning network, termed FSLNets-TLSA, is proposed. This method employs data preprocessing techniques to enhance the temporal semantic logic features of the text. The network architecture consists of an encoder, an induction module, and a relation module, which aim to capture the implicit temporal logic semantic information in the input text. In addition, an enhancement module is incorporated to improve the accuracy of monitoring semantic recognition. The effectiveness of the proposed method is validated through experimental evaluations conducted on three public datasets comprising a total of 3 533 samples, and a comparison with similar tools. The analysis demonstrates an average Accuracy, Recall, and $F1$-score of 96.55%, 96.29%, and 96.42%, respectively.

Key words: temporal logic (TL); semantic analysis; formal specification

---

* 基金项目: 陕西省重点研发计划 (2023-YBGY-229); 西安市科技项目 (22GXFW0025)
收稿时间: 2024-12-13; 修改时间: 2025-01-21; 采用时间: 2025-02-11; jos 在线出版时间: 2025-06-11
CNKI 网络首发时间: 2025-06-11

时序逻辑 (TL) 是一种用于描述和推理系统状态随时间变化的逻辑语言, 用于明确指定时序扩展任务, 通过时序运算符增强了标准命题逻辑的传统概念, 这些运算符能够表达系统在不同时间点上的性质和行为. 根据时间模型和语法特性的不同, TL 可以被更加详细地划分为线性时序逻辑 (LTL)[1]、计算树逻辑 (CTL)[2]和命题投影时序逻辑 (PPTL)[3]等形式语言. 得益于其强大的表达能力, 时序逻辑已被广泛应用于多个计算机科学和工程领域, 确保动态系统在整个运行过程中满足某种特定的时序约束和行为要求, 尤其在形式化验证[4−7]、机器人系统控制[8−10]和需求规范化定义[11]等方向. 其中 PPTL 较其他时序逻辑具有 3 个方面优势, 首先 PPTL 的表达能力完全正则等价于μ-演算, 且强于 LTL、CTL, 目前已证明 PPTL 是可判定的. 其次 PPTL 可以应用于多核系统建模, 其表达监控语义的投影结构与其他时序逻辑语言相比更直观, 也更易于处理多核并行计算实例. PPTL 也能够对监控系统中的多级中断需求进行建模. 最后, PPTL 长期以来一直面向安全关键系统建模, 特别是针对监控需求建模和多核并行计算分析, 其在业界得到广泛的应用, 如货币交易系统、安全关键任务调度系统、区块链系统、交通信号灯控制系统、社交网络监控、网络入侵检测等[12−14].

尽管 PPTL 在刻画复杂任务行为方面具有优势, 但对于非专家用户来说难以掌握使用[15]. 以形式化验证在工业场景下的实际应用为例, 待验证的性质规约通常来源于软件系统的需求描述文档. 一方面, 缺乏数学背景的领域工程师难以准确理解 PPTL 复杂的时序逻辑语义; 另一方面, 人工将需求文档中的需求描述语句改写为时序逻辑任务的效率低下, 阻碍了形式化规约在工业领域的推广应用. 因此, 研究一个可以将 PPTL 转化为时序逻辑公式的方法具有重要意义, 但隐含时序语义无法识别造成转化成功率低的问题还未解决, 如 Zhang 等人[16]提取的针对智能家庭物联网的生成 LTL 规范方法和 ICON[17]等方法均出现难以推断隐含时间约束, 而导致生成 LTL 规范准确率变低的问题. 因此识别自然语言时序语义是重要的, 其可以辅助和推动生成时序规范生成工作进行.

早期的语义解析工作大多采用基于规则和模板的方法[18], 难以识别文本中隐含的时序关系, 缺乏灵活性和泛化能力, 实际应用效果较差. 后续基于统计的方法被研究出来, 如隐马尔可夫模型[19], 通过隐状态建模文本结构, 利用统计概率解析文本语义, 然而其准确性仍受限于需求样本规模. 目前神经网络及大模型的发展[20], 可以更好地捕捉文本潜在的语义信息, 这推动了隐含语义识别工作, 然而基于神经网络的方法应用于时序逻辑语义分析仍然存在着诸多局限性, 本文技术面临如现有需求文本数据集规模不足, 语义分析不够准确等一般性问题, 具体在时序逻辑语义研究中体现的主要难点如下.

1) 部分需求文本存在时序逻辑语义模糊性的问题, 准确识别其隐含语义是困难的.

2) 需求文档的冷启动问题, 需要应对需求样本中存在的时序标签样本稀疏和不平衡的问题.

3) 面对监控语义分析[21]准确率较低的问题.

但又区别于一般的基于神经网络的语义解析任务, 本文在识别 PPTL 的语法和语义具有其独特技术挑战. 在数据预处理上, 其需要采用特定技术来突出具有时序依赖关系的特征信息. 在网络结构设计上, PPTL 公式的严格性要求网络结构更倾向于保留句子的完整时序逻辑信息, 而非只强调文本局部特征信息. 同时逻辑公式中复杂时序标签通常由多种简单时序逻辑关系排列组合构成, 这就要求网络结构必须能捕捉并强调不同网络层和神经元之间的特征信息的相关性. 此外, 语义分析结果的评估计算必须精确量化时序依赖和逻辑一致性, 以确保标签分类的正确性. 在复杂稀疏的语义识别上, 需要引入能保持语义一致性、保留时序结构的数据增强模块, 保障其语义识别准确率. 这些特殊性符合 PPTL 时序逻辑的严格要求, 保障网络结构设计严谨有效.

针对上述问题, 本文提出了基于小样本学习的神经网络 (FSLNets-TLSA) 来实现时序逻辑语义分析方法. 虽然以 PPTL 为对象开展语义分析工作, 但该工作对 LTL、CTL 等时序逻辑语义分析工作存在迁移性和泛用性. FSLNets-TLSA 由 3 个功能模块组成. 在数据预处理模块, 对需求文本进行数据清洗降低噪声, 并使用命名体识别和依存关系分析提取特征信息向量, 用来突出文本时序逻辑语义和应对自然语言文本描述模糊性问题. 在小样本神经网络模块, 其输入由文本向量串联特征信息向量组成, 其网络结构由编码器, 归纳模块和关系模块堆叠构成, 用来捕捉文本中隐含的时序逻辑语义信息, 同时在编码器中引入自注意力机制增加对长文本语义的理解能力. 在模型增强模块, 为了适应更复杂语义识别, 又结合同义词替换和回译的数据增强方法和对抗训练方法, 用来提高时序逻辑分析的准确率, 以及增强神经网络模型鲁棒性. 在公开数据集上的实验结果和相似工具的对比结果证明了本文方法

的有效性. 本文基于之前工作[22]完善, 旨在识别需求文本中体现的核心时序逻辑算子, 而非直接生成完整的时序逻辑规范. 隐含语义无法有效解析通常会导致自然语言转化成逻辑公式准确率下降, 而本文方法通过明确关键时序逻辑标签, 为形式化公式生成中的转化模板选择与方案设计提供了理论依据和实质支持, 辅助 Zhang 等人[16]进行形式化公式生成的相关研究, 提升其规约生成的准确性与可靠性.

本文第 1 节介绍语义分析的相关方法和研究现状. 第 2 节介绍本文所研究的 PPTL 基础知识. 第 3 节介绍本文构建的神经网络结构模型. 第 4 节通过对比实验验证所提模型的有效性. 最后总结全文并讨论未来研究方向.

1 时序语义分析相关工作

语义分析是自然语言理解领域的一个重点方向, 其主要目标是解析和理解自然语言的涵义, 为机器解读自然语言提供理论基础. 语义分析的应用场景广泛, 包括但不限于机器翻译及文本分类等. 在自然语言处理 (NLP) 的早期阶段, NLP 提供多种语法语义解析方法, 辅助计算机理解与处理文本, 研究者们可以依赖基于规则和模板的方法来进行语义分析[17]. 然而由于自然语言的高度复杂性和多变性, 这些方法在实际应用中存在泛用性过低的局限性. 随着计算能力的增强, 研究逐渐转向采用基于统计的方法, 如隐马尔可夫模型和最大熵模型, 这些模型依赖于大规模语料库来实现词性标注和有效的语法分析. 进入 21 世纪, NLP 领域经历了技术的革命性变革, 特别是机器学习和深度学习的广泛应用, 极大地推动了语义分析技术的发展, 尤其是神经网络的应用, 包括但不限于循环神经网络 (RNN)、长短时记忆网络 (LSTM) 以及更为先进的变换器模型 (Transformer), 这些深度学习框架在处理自然语言的深层语义理解方面显示了卓越的性能, 但其表现仍然受限于数据集的规模和模型的过拟合问题. 最近几年, 预训练模型和大模型发展如双向 Transformer (BERT)[23]、生成式预训练 Transformer (GPT)[24]和 XLNet[25]等, 这些模型主要基于深度神经网络和自注意力机制, 通过大规模无监督预训练和任务特定微调, 实现高效的自然语言理解与生成, 已成为推动语义分析技术进步的关键. 这些模型通过在大规模文本数据上进行预训练, 不仅学习到了丰富的语言表征, 而且在语义分析中展示了优异的性能. 时序语义分析是语义分析的一个重要研究方向, 通过整合自然语言文本中的词义、句法结构和文本内容, 识别文本蕴含的时序逻辑, 为自然语言转为形式化公式并应用于后续的形式化验证工作提供重要的技术支持.

目前语义分析有 4 个主流方向: 1) 基于规则和模板的方法. Blasi 等人[26]开发的 CallMeMaybe 方法可以自动识别 Java 类的时序逻辑约束, 并指导测试用例生成器遵守时序约束执行方法调用序列, 它能够从 JavaDoc 注释中自动识别 Java 类的显式和隐含的时间约束, 并使用依存关系分析和预定义模板生成有关的时序规范. 2) 基于统计的方法. Zhong 等人[19]开发了 Doc2Spec 方法从 API 文档中提取时序语义规范, 该方法在文档中提取函数的描述和继承关系, 然后在从函数描述中建立动作-资源对, 并通过隐马尔可夫模型分析得到的动作-资源对和继承关系, 借此推断出各个类或接口的资源使用规范. 3) 基于神经网络的方法. Pandita 等人[17]开发了 ICON 方法, 通过结合机器学习和 NLP 来识别和推断时序约束信息. 它可以依据词组、句法结构和语义特征自动学习句子模式, 有效克服了仅依靠关键词搜索识别文本时序逻辑语义的局限性. 此外为了准确识别隐含语义, 它结合 API 文档和通用英语词典系统创建了专门的领域词典, 不仅提高了语义分析的准确性, 有效地从文档中推导出形式化规范. WHYPER 方法[27]采用语义模型来关联手机应用市场中的 API 描述与相应的权限需求, 实现从描述文本到权限规范的映射. 该方法运用 NLP 技术和语义图进行深度语义推理, 借此推断出应用程序对权限的需求. 4) 基于大模型的方法. Fuggitti 等人[20]使用 NLP 和大语言模型开发了 NL2LTL 工具, 使自然语言指令能够被转换成 LTL 公式, 其时序语义分类结果表现在选用不同的预定义的自然语言解析模板过程中.

现有方法识别时序逻辑语义分析仍存在挑战. CallMeMaybe 方法尽管能适应多种文档风格的代码项目, 但它过度依赖注释的完整性和准确性, 语义模糊的注释可能导致时间约束的识别不准确, 因此无法应对自然语言需求文本语义模糊性的问题. Doc2Spec 方法针对特定领域文档设计了模板, 但并未设计对复杂时序逻辑有效的模板, 因此其固定的模板限制了其广泛适用性. ICON 方法虽利用机器学习和启发式方法, 但它无法有效推断出隐含时间约束, 并且将分析隐含时间约束的工作推迟到了未来的计划中. WHYPER 方法依赖于语义模型, 但缺乏有效的

数据预处理步骤. 如果自然语言文本存在语义模糊性和序列过长的问题, 可能会导致对权限需求的分析不准确. NL2LTL 基于大模型分析语义, 在处理复杂语境或模糊表达时可能出现错误的分析和解释. 因此, 本文基于神经网络的方法, 综合多种 NLP 技术来实现数据预处理工作, 在文本中提取特征来强调时序语义信息, 同时构建有效的小样本神经网络结构, 克服了识别隐含和复杂的语义的难点, 并加入模型增强模块提高模型的鲁棒性和泛用性, 克服了时序标签样本的稀疏和不平衡问题.

2 基础知识

本文所提方法主要基于 PPTL 和监控语义, 具体给出了 PPTL 的语法和语义, 并介绍了自然文本的时序语义分类类型, 即时序操作符作为研究内容的理论基础, 最后也给出了监控语义的定义和解释. 下面就相关概念和基本知识予以介绍.

2.1 PPTL 语法

令 $Prop$ 表示可数的原子命题集合, PPTL 通过公式 (1) 给出语法定义:

$$P = p|\neg p|P_1 \wedge P_2|\bigcirc P|(P_1,\ldots,P_m)\ prj\ P \tag{1}$$

其中, $P \in Prop$ 的原子命题, $(P_1,\ldots,P_m)$ 和 P 表示 PPTL 公式, *next* ($\bigcirc$) 和投影 (*prj*) 是两个基本的时序操作符. 依据 PPTL 公式是否包含时序操作符, 可以划分为两大类: 一类是时序公式, 一类是状态公式.

2.2 PPTL 语义

PPTL 的状态定义: 使用字母 s 表示一个区间状态. 接着引入 p, $p \in Prop$. 用大写字母 B 表示真值集合, 该集合包了两个元素: $B = \{\text{true}, \text{false}\}$, 分别代表真和假. 在状态 s 上, p 的真值是由从 s 到 B 的映射确定, $s[p] = \text{true}$ 就意味着状态 s 下, p 的值为真. 对于状态 s 上的任意原子命题, 其真值公式定义为公式 (2):

$$s : Prop \to B \tag{2}$$

PPTL 中的区间定义: 定义区间 σ 为非空状态序列, 表示为 $\sigma =< s_0, s_1, \ldots, s_{|\sigma|} >$, 并且依据状态个数可将区间状态划分为两种类型: 有穷区间和无穷区间. 有穷区间长度 $|d| = c - 1$, 其中 c 表示区间内状态个数. 无穷区间长度 $|d| = \omega$, 其中 ω 表示无穷大. 为了统一区间长度描述, 将非负整数集 N_0 扩展为 $N_\omega = N_0 \cup \{\omega\}$. 在 N_ω 中, $\omega = \omega$, 对于 $\forall i \in N_0$, 将 N_0 中的操作符 $<$、$\leqslant$ 和 $=$ 扩展到 N_ω 中, 并定义操作符 $\preceq$ 为 $\leqslant - \{(\omega, \omega)\}$. 定义区间 σ 的子区间使用 $\sigma_{m\ldots n}$ $(0 \leqslant m \preceq n \leqslant |\sigma|)$ 表示.

PPTL 中的投影定义: 定义一个连续的非负整数序列 $r_1,\ldots,r_n$, 以及一个区间 $\sigma =< s_0, s_1, \ldots, s_{|\sigma|} >$, 其中 $n \geqslant 1$ 且 $0 \leqslant r_1 \leqslant \ldots \leqslant r_n \preceq |\sigma|$. 为了表示投影操作, 引入辅助算子 $\downarrow$. 那么区间 σ 投影在序列 $r_1,\ldots,r_n$ 上的结果如公式 (3) 所示, 其中 $t_1, t_2, \ldots, t_l$ 为 $r_1,\ldots,r_n$ 删除重复项所得严格递增序列:

$$\sigma \downarrow (r_1,\ldots,r_n) =< s_{t_1}, s_{t_2}, \ldots, s_{t_l} > \tag{3}$$

2.3 时序操作符

基本操作符的定义与经典一阶逻辑中的定义相同, 例如 true, $\vee$, $\to$, $\neg$, 并且对任意公式都有 $\text{true} \stackrel{\text{def}}{=} P \vee \neg P$、$P_1; P_2 = (P_1, P_2)\ prj\ \varepsilon$ 和 $\varepsilon \stackrel{\text{def}}{=} \neg\bigcirc\text{true}$. 以下分别介绍本文用到的时序操作符:

$$\begin{cases} A1\ \ sometimes\ (\Diamond):\ \Diamond P \stackrel{\text{def}}{=} \text{true}; P \\ A2\ \ always\ (\Box):\ \Box P \stackrel{\text{def}}{=} \neg\Diamond\neg P \\ A3\ \ next\ (\bigcirc):\ \bigcirc^n P \stackrel{\text{def}}{=} \bigcirc(\bigcirc^{n-1} P)(n > 0) \\ A4\ \ implication\ (\to): P \to Q \stackrel{\text{def}}{=} \neg P \vee Q \\ A5\ \ prj: (P_1,\ldots,P_m)\ prj\ Q \end{cases},$$

其中, $A1$ 公式在区间 $\sigma =< s_0, s_1, \ldots, s_{|\sigma|} >$ 上成立且仅当存在某个子区间 $\sigma_{(i\ldots|\sigma|)}$ $(0 \leqslant i \preceq |\sigma|)$ 使得 P 成立, $\Diamond P$ 语义表示如图 1(a) 表示; $A2$ 公式在区间 σ 上成立当且仅当 P 在所有子区间上均成立. $\Box P$ 的语义如图 1(b) 所示; 公式在

区间 σ 上成立, 当且仅当 σ 的长度大于 0 并且在子区间 $\sigma_{(i...|\sigma|)}$ $(0 \leqslant i \prec |\sigma|)$ 上成立, $\bigcirc P$ 的语义如图 1(c) 所示; *A*4 公式是基本操作符; *A*5 公式的投影操作符 *prj* 是 PPTL 的核心操作符, 它可以描述进程间并发和同步操作. 如图 2 是投影公式 (P_1,P_2) *prj* Q 的语义表示. P_1 和 Q 在状态 t_0 同时开始执行. 在整个过程中, Q 和 $P_1;P_2$ 在一段时间内并行执行, 但可能存在 3 个不同的时间点结束, 3 种可能的结果分别为: (1) Q 在 P_2 之后结束, (2) P_2 和 Q 同时结束, (3) Q 在 P_2 之前结束.

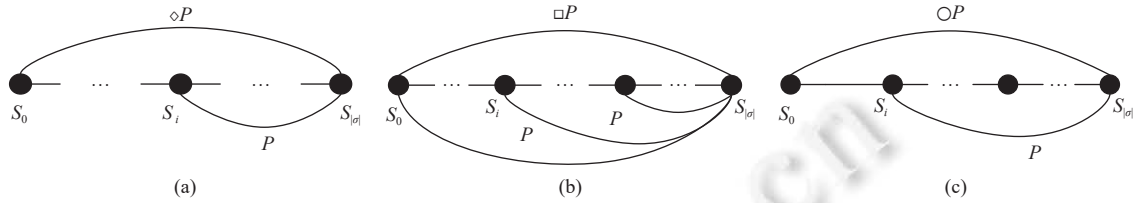


图 1 公式 $\diamond P$、$\Box P$ 和 $\bigcirc P$ 的语义

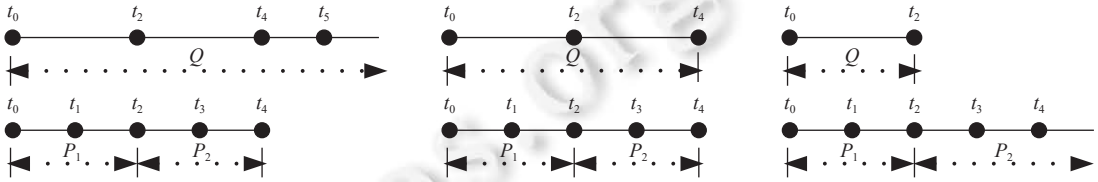


图 2 公式 (P_1,P_2) *prj* Q 的语义

2.4 监控语义

监控语义是在运行时验证 (runtime verification, RV) 中, 系统通过实时监控其行为并根据预定协议及时反馈, 以检测错误或违规情况. 通常, 运行时验证利用形式化规范语言来定义安全性质 φ, 当使用 PPTL 来描述监控语义时, 首先定义系统的期望性质, 并将相应的 PPTL 公式转换为监控器, 即一个有限状态自动机 (finite state machine, FSM). 传统 PPTL 使用二值语义来进行判定, 输出结果为 true 或 false. 然而, 由于系统在执行过程中会产生无限状态的轨迹, 并且在没有外界干扰的情况下是非终止的, 运行时验证无法获取系统的完整执行轨迹, 因此无法用二值语义来全面描述其性质. 为了应对这一问题, 引入一个额外的真值 *inconclusive* 扩展 PPTL 的值域, 其定义[21]如下:

$$[u \models \varphi] = \begin{cases} \text{true}, & \forall \omega \in \sum^{\omega} : u\omega \models \varphi \\ \text{false}, & \forall \omega \in \sum^{\omega} : u\omega \not\models \varphi \\ \textit{inconclusive}, & \text{otherwise} \end{cases} \tag{4}$$

其中, $\sum$ 表示字母表. 对于一个性质 φ 和一个有限前缀 u, 如果 u 的所有扩展都满足该性质, 则说明 u 满足 φ; 如果 u 的所有扩展都违反了性质, 则说明 u 违反了 φ; 否则, 它无法说明 u 是否满足或违反了性质 φ.

语法语义的定义为数据预处理及结果准确性评估做了理论铺垫, 数据预处理中筛选的重点携带时序逻辑语义的单词来源于定义中时序逻辑符号. 依存关系提取的时序逻辑依赖关系, 也来源于定义中的时序逻辑关系, 数据预处理实现了有效的特征信息提取, 从而保障了神经网络分类准确的性能. 此外, PPTL 的语义识别的正确性评估遵循严格的数理逻辑定义, 其通过高阶张量运算构建的时序标签向量与时序符号定义应具有语义一致性, 确保网络能够计算精确的关系得分, 得到需求文本的真实时序逻辑语义分析结果.

3 基于小样本网络的时序逻辑语义分析 FSLNets-TLSA

鉴于深度神经网络强大的表示能力和特征提取能力, 本文提出一种基于小样本网络的时序逻辑语义分析方法, 该方法的整体架构如图 3 所示.

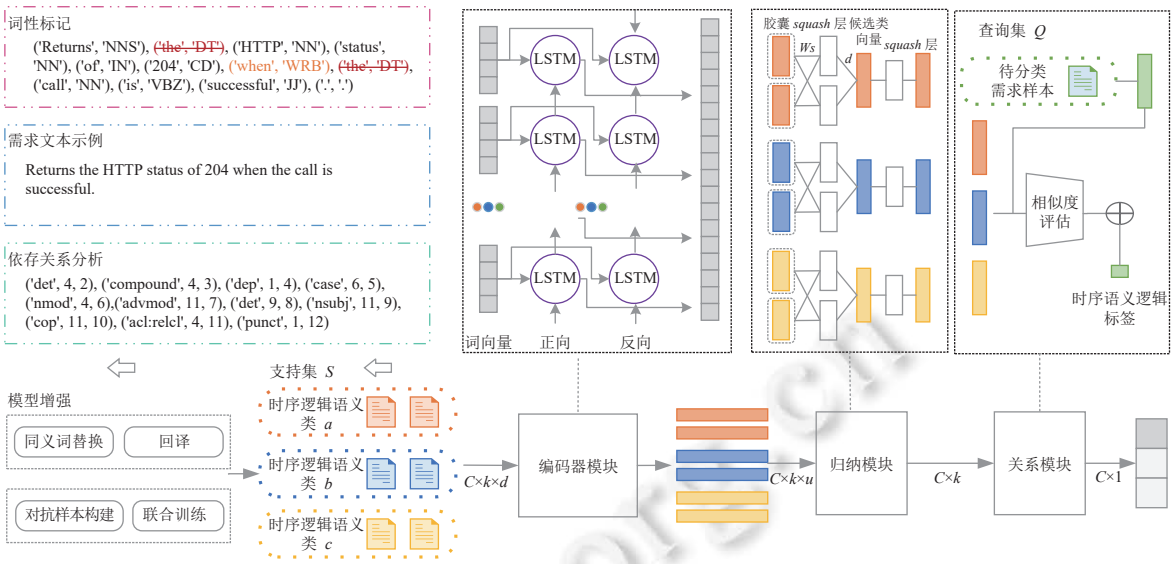


图 3 FSLNets-TLSA 整体框架图

本文采用了小样本神经网络技术和 NLP 相结合的方法, 模型由数据预处理模块、编码器模块、归纳模块、关系模块和数据增强模块组成. 在数据预处理阶段, 通过数据清洗降低需求文本中的噪声, 并使用词性标注来收集带有时序标签的特征信息, 同时删除一些信息熵较低的单词. 此外, 利用依存关系分析推断出句子中单词之间的语义关系, 进而提取单词之间的关联性特征信息. 以上所提取的特征信息被用于突出文本时序逻辑语义特征信息, 用以应对自然语言文本描述模糊性问题. 在网络训练时, 网络结构由编码器, 归纳模块和关系模块堆叠构成, 用来捕捉文本中隐含的时序逻辑语义信息. 在编码器中引入自注意力机制, 通过计算序列中所有词之间的加权关系, 使模型能够捕捉远距离依赖关系, 从而增强对长文本语义的理解能力. 在归纳模块应用动态路由算法共享样本特征信息, 捕捉不同神经层和神经元相关性, 从而提升输出和真实预测结果的耦合系数. 为了适应更复杂的监控语义识别, 使用同义词替换和回译进行模型增强, 同时集成对抗训练模块, 提高神经网络模型的鲁棒性和语义分析的准确率.

3.1 网络输入模块

本模块的数据来源于需求文档中原始的需求文本, 这些文本可能存在如语义模糊性、书写格式不统一、停用词过多、文本样本标注不平衡、文本规模过少等缺陷, 本模块旨在通过数据预处理和小样本数据集构建方法, 提取时序逻辑特征信息并构建良好的数据表示, 最终输出需求文本的特征向量表示.

3.1.1 数据预处理

为了降低数据噪声, 本论文使用词性标注 (part-of-speech tagging, POS) 分析单词词性, 当输入的自然文本句子表示为 $V=\{v_1,v_2,v_3,\ldots,v_T\}$, 其中 T 是句子中单词的数量, 时序标签的集合可以表示为 $TL=\{tl_1,tl_2,tl_3,\ldots\}$.

通过 POS 标记后的句子可表示为:

$$tl_i=\Gamma(v)\in TL \tag{5}$$

其中, $\Gamma(v)$ 本文采用 NLTK 词性标注工具, 也可采用 Stanford CoreNLP 等工具, 其中表 1 中所列标签都为重点携带时序逻辑语义的单词标注, 本文提取这 4 种单词作为时序特征信息补充在文本向量之后. 同时依据标记结果移除需求文本中干扰性信息, 特别是移除在文本中频繁出现但通常不携带重要信息的常见 dt-停用词, 比如 a, the. 无用的单词标签被定义为 $\{ul_1,ul_2,ul_3,\ldots\}$, 数据降噪过程可表示为:

$$V'=(v^+\kappa^+)^+\wedge\neg(Delete(v)\wedge\Gamma(v)\in\{ul_1,ul_2,ul_3,\ldots\}) \tag{6}$$

其中, $Delete(v)$ 表示删除单词操作, + 表示一个到多个的关系, κ 表示标点符号. 最后, 通过依存关系分析, 确定文本

中单词之间的语义关联关系. 给定句子 V 中, 使用 $(v_i, v_j, \gamma_{i,j})$ 来描述句子单词之间的依赖关系, 此关系可以捕捉到单词间语法和语义的连接.

表 1 词性标注中的时序逻辑语义提取

$\Gamma(v)$	解释
vb-动词	不同时态下的动词
rb-时间修饰语	表达时间 (“立即”等)或逻辑顺序 (“因此”等)的副词
cc-连词	表达句子或短语间的逻辑关系 (“和”等)的连词
in-介词	表达时间顺序 (“在…之前”“直到”等)或者逻辑关系 (“因为”等)的介词

表 2 所列依存关系是存在时序逻辑依赖的连接, 需求文本若存在以上连接, 则将其作为关联性特征信息补充到文本向量之后. 以上的数据预处理步骤, 可以突出输入需求文本的时序逻辑语义特征信息, 减少文本的噪声信息. 最终将文本向量与特征向量串联构成小样本网络的输入数据, 有利于应对自然语言文本描述模糊性问题.

表 2 依存关系中的时序逻辑语义提取

$(v_i, v_j, \gamma_{i,j})$	解释
advcl-副词性修饰语	这种关系可以表示副词修饰动词或其他谓词, 可能包含了时间上的逻辑关系, 通常为时间状语从句
mark-从句引导词	这种关系可以表示if引导的从句
tmod-时间修饰语	这种关系表示表示时间的名词短语, 表示动作发生的时间
advmod-副词修饰语	这类通常用于修饰形容词或动词的副词, 包含表示时间的副词, 如(“现在…”“…之后”等)含时序信息

3.1.2 小样本数据集构造

本节继续将预处理后的输入数据划分为 3 个子集, 训练集 C_{train}、测试集 C_{test} 和验证集 C_{dev}, 在训练集中每个支持集中的类别仅包含 k 个标记样例, 由于 k 值设置较小, 难以直接训练有效的监督模型. 为此, 本文采用元学习[28]的思想, 从 C_{train} 中提取可转移的知识, 构造有效的支持集和查询集. 这样构造模型的输入, 可以使模型在标签受限的情境中表现出更强的适应性和学习能力, 具有更好的泛化和快速适应新任务的能力, 从而显著提升小样本技术性能. 本文所设计的小样本网络的目标是学习给定的支持集 S, 以最大程度地减少在查询集 Q 上的损失, 其详细步骤在算法 1 中呈现.

算法 1. 元学习算法.

输入: 包含有标签的样本训练集 C_{train}, 初始支持集 S, 初始查询集 Q;
输出: 更新模型参数.

1. Repeat N times:
2. 从训练集的类别空间中随机选取 C 个类别
3. 在 C 个类别中, 随机选取 k 带标签样本作为支持集 S, 其中 $S=(x_s,y_s)_{s=1}^{m}(m=k\times C)$
   并从剩余样本中随机选 n 个样本组成查询集 Q, $Q=(x_q,y_q)_{q=1}^{n}$
4. 将支持集 S 输入模型, 并最小化查询集 Q 的损失, 用于更新模型参数
5. End Repeat

3.2 神经网络架构模块

本模块接收需求文本的特征向量表示作为输入, 经过网络的迭代优化训练后, 最终模型收敛, 输出需求文本所蕴含的时序逻辑语义分类标签.

3.2.1 编码器模块

编码器模块采用了具有自注意力机制能力的双向递归神经网络, 从给定的输入文本 V, 得到文本向量化嵌入表示. 输入被定义为 $x=\{x_1,x_2,\ldots,x_i\}$, 其维度为 d, 其标签为 y, 然后使用双向的 LSTM 提取嵌入表示 e, 网络结构

如公式 (7) 所示:

$$h_i = \eta(W \cdot [h_{i-1}, x_i] + b) \odot \tanh(Z_i) \tag{7}$$

其中, η 为输出门的 *Sigmoid* 激活函数, $\odot$ 是逐元素乘积, W 和 b 是输出门的权重和偏置. Z_i 是记忆单元. 在反向 LSTM 中, $\bar{h}_i$ 信息按照序列 x_i 到 x_1 的反向顺序处理. 双向 LSTM 的每个时间步 i 的输出是 e, 通过加权系数 a 拼接 LSTM 的隐藏状态得到, 即 $e = \sum_{j=1}^{i} a_j * h_j$ ($*$ 为标量乘法). 编码器可以同时利用过去和未来的上下文信息, 更有利于捕捉需求长文本的语义. 本模块将输入的时序特征进行编码, 使用 LSTM 结构保留其时序特征. 此编码器与胶囊网络和关系模块紧密结合, 使得编码后的时序信息能够在后续的关系计算和分类过程中持续发挥作用, 通过对时序特征的精准编码, 也保证了高阶需求文本和低阶向量文本的语义一致性要求.

3.2.2 归纳模块

该模块主要由胶囊网络和动态路由算法组成. 胶囊网络是一种改进于传统 CNN 的深度学习架构, 其使用矢量方向和模长共同表示特征及状态, 这种结构更适用于保留编码后的时序逻辑信息, 这对于 PPTL 时序标签分类尤为重要. 动态路由算法允许胶囊网络在特征层次之间动态调整连接权重, 这种动态性使得网络能够更灵活地捕捉和强调时序特征在不同网络层和神经元之间的相关性, 使得网络能够更精准捕获到时序逻辑排列组合信息, 识别和区分复杂时序依赖.

根据编码器模块得到嵌入表示 e, 从支持集 S 中获得的样本向量为 e^s, 查询集 Q 中获得的向量作为查询向量 e^q, 那么从样本向量 $e^s_{i,j}$ 到 one-hot 类向量 Υ_i 的非线性映射可表示为:

$$\{e^s_{i,j} \in R^{2u}\}_{i=1,\cdots,C,j=1,\cdots,k} \mapsto \{\Upsilon_i \in R^{2u}\}_{i=1}^{C} \tag{8}$$

LSTM 的隐藏状态 h_n 的大小为 u, 隐藏状态中门控单元的权重矩阵决定了 x_i 和 u 之间的映射关系. 对每一个类别的样本应用胶囊网络中的动态路由算法[29], 每个类别中的胶囊将负责捕捉该类别的特征, 并通过动态路由算法完成信息传递. 为了使模块能够支持不同规模的输入, 支持集中的所有样本向量共享相同的变换权重 $W_s \in R^{2u\times 2u}$. 其中每个样本的预测向量 $\tilde{e}^s_{i,j}$ 的计算公式可表示为:

$$\tilde{e}^s_{i,j} = \vartheta(W_S e^s_{i,j} + b_s) \tag{9}$$

其中, ϑ 是非线性压缩函数 *squash*, 它可以使向量方向保持不变, 但是压缩了向量大小. 然后, 模块继续将样本嵌入向量映射到对应的类向量, 动态路由算法设置了 ϖ_i 是耦合系数的 logits 值, 该系数确定了不同胶囊之间的关系和联系强度, 动态路由算法可在每次迭代过程中自动调整连接强度. 通过 *Softmax* 归一化路由机制, 确保每个类别 i 与该类中所有支持样本之间的耦合系数 $d_i = Softmax(\varpi_i)$ 的总和为 1. 接下来, 每个类别样本的预测结果是类 i 中所有样本预测向量的加权和:

$$\widetilde{\Upsilon}_i = \sum_j d_{i,j} \cdot \tilde{e}^s_{i,j} \tag{10}$$

使用非线性压缩函数得到路由过程的矢量输出 $\hat{\Upsilon}_i = \vartheta(\widetilde{\Upsilon}_i)$, 确保其长度不会超过 1. 每轮迭代的最后, 使用一致性路由方法调整耦合系数的值 $d_{i,j} = Softmax(\varpi_{i,j} + \tilde{e}^s_{i,j} \cdot \hat{\Upsilon}_i)$. 当生成的类别预测向量在某一样本预测呈现较高的标量输出时, 会触发自上而下的反馈, 更新该样本的耦合系数, 并相应地降低其他样本的耦合系数, 这种调整策略增强了小样本学习的分类准确率. 根据以上描述, 本文将其总结到了算法 2 中进行展现.

算法 2. 动态路由算法.

输入: 支持集 S 中的样本向量 $e^s_{i,j}$, 初始化耦合系数的对数 $\varpi_{i,j} = 0$;
输出: 返回预测结果 $\hat{\Upsilon}_i$.

1. 计算所有样本的预测向量;
2. $\tilde{e}^s_{i,j} = \vartheta(W_S e^s_{i,j} + b_s)$
3. Repeat N times:

4. $d_{i,j} = Softmax(\varpi_{i,j})$

5. $\widetilde{\Upsilon}_i = \sum_j d_{i,j} \cdot \tilde{e}^s_{i,j}$

6. $\hat{\Upsilon}_i = \vartheta(\widetilde{\Upsilon}_i)$

7. 更新所有样本:

8. $d_{i,j} = Softmax(\varpi_{i,j} + \tilde{e}^s_{i,j} \cdot \hat{\Upsilon}_i)$

9. End Repeat

10. Return $\hat{\Upsilon}_i$

3.2.3 关系模块

在关系模块中, 使用归纳模块生成的预测结果 $\hat{\Upsilon}_i$ 和通过编码器模块得到的 e^q 查询向量来评估每个查询 q 和类之间的相关性并生成关系得分, 其取值范围在 0–1 之间, 本文加入了神经张量层[30]作为交换函数, 并定义了关系向量的计算公式如下:

$$\chi(\hat{\Upsilon}_i, e^q) = f(\hat{\Upsilon}_i^{\mathrm{T}} M^{[1:g]} e^q) \tag{11}$$

其中, $M^{[1:g]}$ 是神经张量层, M^g 是指索引 g 下的神经网络层, $\hat{\Upsilon}_i^{\mathrm{T}}$ 是 $\hat{\Upsilon}_i$ 的转置, $M^g \in R^{2u \times 2u}$. $f(\cdot)$ 是 ReLU 非线性激活函数, 最终第 i 类和第 q 查询之间的关系最终得分 r_{iq} 由 η 函数激活的全连接层得出:

$$r_{iq} = \eta(W_r \chi(\hat{\Upsilon}_i, e^q) + b_r) \tag{12}$$

神经张量层通过对查询与类之间的时序依赖关系计算, 确保网络能够生成精确的关系得分.

3.2.4 目标函数

本文采用的是均方差损失函数, 该函数不仅计算样本的真实标签预测损失, 还加入了其余标签的预测损失. 均方差损失函数通常会产生较平滑的梯度, 对偶然样本误差不那么敏感, 更有效地优化模型的预测概率, 使其输入样本更接近真实的标签分布. 目标函数将关系模块得到的最终得分 r_{iq} 回归到真实得分 y_q 进行训练. 在回归过程中, 设定匹配正确的相似度为 1, 匹配错误的相似度为 0, 使得模型能更好地理解样本之间的相似性关系, 模型的目标损失函数定义为:

$$Loss(r_{iq}|S,Q) = \sum_{i=1}^{C} \sum_{q=1}^{n} (r_{iq} - 1 * (y_q == i))^2 \tag{13}$$

其中, $*$ 表示逐元素乘, n 为查询集大小, 在算法 1 已定义, $y_q == i$ 的取值为 0 或者 1, C 为预测类别总数. 依据元学习算法, 减少目标函数损失计算神经网络权重梯度, 调整权重让网络更好地学习到数据的时序逻辑特征信息.

本文所提网络结构通过多层次的特征表示和逐层的合成过程, 确保时序特征在每一层都得到保留和强化. 通过胶囊网络的多维表示和关系模块的高阶交互计算, 网络能够逐步合成出符合时序逻辑的最终分类结果.

3.3 模型增强模块

相较于其他时序语义分类, 数据集中监控语义的自然语言需求数量极少, 这可能导致 FSLNets-TLSA 网络在训练到足以正确分析监控语义时陷入过拟合, 进而在分析新数据时性能下降. 为了应对这一问题, 本文引入并改良了 EDA 数据增强[31,32]和 PGD 对抗训练[33]两个模块, 旨在增强网络模型的性能. EDA 数据增强通过同义词替换、随机插入、删除和交换等方式提升文本数据的多样性, 而 PGD 对抗训练通过在输入数据上生成对抗扰动来增强模型的鲁棒性, 从而提高对抗攻击的防御能力. 融入增强模块后, FSLNets-TLSA 扩充形成 FSLNets-MSA (few-shot learning networks-monitoring semantic analysis) 网络以提升模型对监控语义的识别能力. 具体而言, 本文在输入层后通过同义词替换和回译方法扩充数据集, 以缓解数据稀缺问题. 在词向量层实施 PGD 对抗训练, 以增强模型的鲁棒性.

本文采用同义词替换和回译两种 EDA 数据增强策略, 以增加数据的多样性并提升模型的泛化能力. 同义词替换通过将原始数据中的词语替换为其同义词来生成新句子, 扩大数据集规模同时保持句子原有的语义, 本文使用

的同义词库为 WordNet, 这是一个包含单词间近义关系的人工维护数据库. PPTL 公式对时序语义非常敏感, 任何细微的语义变化都可能导致逻辑上的偏差, 必须严格遵守时序逻辑的要求, 确保替换后句子的逻辑含义保持一致. 尤其是一些特定的逻辑关键字, 如 until、eventually 等, 这些关键字对于表达时间依赖性至关重要, 不可以替换, 本文替换策略在于预定义带有时间信息的实义名词及其对应 WordNet 中同义词都不允许被替换. 回译通过将文本先翻译成其他语言, 再翻译回源语言的方式进行. 本文采用源语言为英语, 翻译语言为法语, 两个语言结构和表达具有较高的相似性, 翻译后不仅扩展了数据集的规模, 也保持了语言之间的语言学联系.

需求时序分类任务要求对抗样本在生成过程中严格控制扰动范围, 确保时序逻辑表达的完整. PPTL 任务中的对抗训练不仅关注语义层面的鲁棒性, 还需要维护需求对时序和逻辑顺序的准确描述. 因此, 本文规定对于输入词的扰动应在 $TL=\{tl_1,tl_2,tl_3,\ldots\}$ 集合之外, 增加该特定的约束条件, 以确保生成的对抗样本不会破坏关键的时序信息.

对抗训练的做法是通过在词嵌入空间中引入扰动以生成对抗性示例, 并将其与原始样本联合训练模型. 这种方法通过改变输入文本的向量表示来影响模型的预测结果, 不仅提高了模型的稳健性和泛化能力, 也减轻模型过拟合的问题. 对抗训练是一个经典的 min-max 问题:

$$\min_{\theta} E_{(x,y)\sim D}[\max_{\|\delta\|\leqslant\varepsilon} Loss(f_\theta(x+\delta),y)],\ \Gamma(x)\notin TL \tag{14}$$

其中, θ 是网络参数, 网络输入由 x 表示, y 是真实标签, D 表示训练集, δ 表示对抗扰动, ε 是对抗扰动的阈值, f_θ 是网络模型. PGD 算法通过在输入数据的梯度方向上引入微小扰动来制造对抗样本, 为了防止扰动的幅度过大, 算法会确保每个生成的对抗样本都限制在一个特定的范围内, 不会超过预设的阈值 ε. 生成对抗扰动可以通过公式 (15) 和公式 (16) 实现:

$$\delta_{t+1}=\prod_{\|\delta\|_F\leqslant\varepsilon}\delta_t+\alpha\frac{g(\delta_t)}{\|g(\delta_t)\|_F} \tag{15}$$

$$g(\delta_t)=\nabla_\delta Loss(f_\theta(x+\delta),y) \tag{16}$$

其中, $\|\cdot\|_F$ 为投影到 Frobenius 范式定义的空间, α 作为扰动更新的步长. δ_t 是第 t 次迭代时的扰动, $g(\delta_t)$ 是当前扰动下的损失梯度. 图 4 详细绘制了基于 PGD 算法的对抗训练过程, 通过执行 T 次梯度上升操作, 逐步形成最终扰动 δ_T, 添加到原始输入文本以构造出对抗样本 x_T. 将对抗样本与原有样本一起训练模型, 能够在保持原有样本可用性的同时, 有效探索潜在的对抗样本空间, 减轻模型过拟合现象.

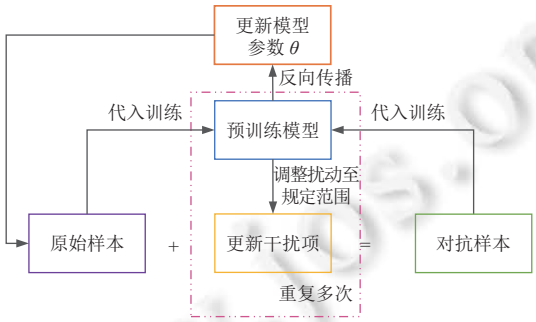


图 4 PGD 对抗训练流程图

4 实验分析

4.1 实验数据

本文在由 Pandita 等人[17]收集的公开数据集上进行实验, 包括 Amazon S3 REST API、Paypal Payment REST API 和 Java.Io.

• Amazon S3 REST API 是基于 REST 架构设计的云存储和检索数据的网络服务接口, 该 API 允许用户以编程方式管理 Amazon S3 中储存数据, 包括上传和下载文件、创建和删除存储桶, 以及管理存储桶中的对象.

• PayPal Payment REST API 是 PayPal 提供的一组用于处理支付交易的编程接口, 该 API 通过 HTTP 请求和 JSON 格式的数据进行通信. 它允许开发者集成 PayPal 支付功能到其应用、网站或服务中, 从而实现在线购物、捐赠和其他支付相关的功能.

• Java.Io 是 Java 编程语言中用于处理输入和输出操作的标准 I/O 库, 此库提供了一组类和接口, 用于与文件、流和其他输入/输出资源进行交互, 主要涉及字符和字节级别的 I/O, 包括文件读写、网络操作、标准输入输出等.

训练集、测试集和验证集的划分比例为 6:2:2, 这种划分比例确保了在小样本学习提供足够的测试和验证数据, 以有效评估和调整模型. 数据集标注情况如表 3 所示.

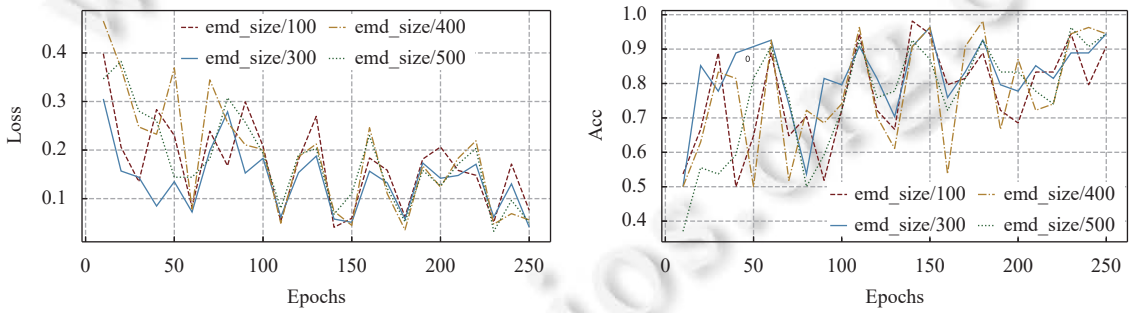
表 3 数据标注实例

标记信息	训练集	测试集/验证集	自然语言句子示例
sometimes (◇)	315	105	If an I/O error occurs
always (□)	477	158	If bMM is null, a NullPointerExceptionMM is thrown
next (○)	163	54	The first byte read is stored into element b[off]MM, the next one into b[off+1]MM, and so on
implication (→)	1 576	525	All objects added to the bucket receive the version ID null
prj	118	42	Schedules the specified task for repeated fixed-rate execution, beginning at the specified time

本文实验环境是一台装有 Ubuntu 22.04.4 LTS 操作系统的主机, CPU 为 Intel(R) Core(TM) i9-14900K, 内存大小为 128 GB, 存储大小为 2 TB 的固态硬盘, 显卡为 NVIDIA GeForce RTX 4090 显卡. 实验设定 LSTM 的隐藏状态大小为 128, 注意力维度为 64. 动态路由算法使用的迭代次数为 3, 关系模块是一个 120 维度的神经层.

4.2 实验验证

本文首先研究了不同维度的单词向量和训练次数对实验结果的影响, 如果两者设置数值过高, 会增加系统的运算复杂度和运行时间. 相反, 如果设置过低可能无法充分捕捉词语间的相似性和类别差异, 从而对模型性能造成负面影响. 实验测试了不同维度词向量和训练轮次, 并研究了它们对模型准确率和损失函数的影响. 相关实验结果展示在图 5 和表 4 中.



(a) 词向量维度对损失函数的影响 (b) 词向量维度对准确率的影响

图 5 词向量维度对模型训练结果影响

表 4 不同 Epoch 下验证准确率对比

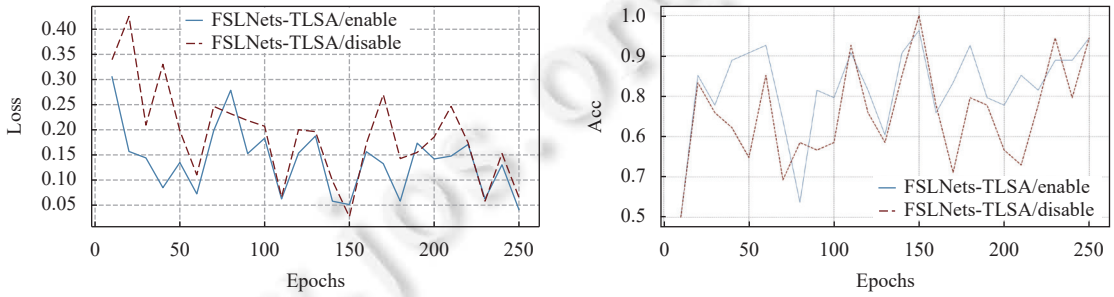
模型/词向量维度	100	200	300
FSLNets-TLSA/100	0.790 476	1.0	0.990 476
FSLNets-TLSA/300	0.723 809	0.714 285	**0.971 428**
FSLNets-TLSA/400	0.628 571	0.980 952	0.828 571
FSLNets-TLSA/500	0.980 952	1.0	0.895 238

实验结果表明, 当单词向量维度 300 时, 训练数据上的图像振荡幅度小、模型稳定性高, 并在训练轮次为 300 时, 验证集上分类准确率表现较好. 在接下来实验中本文将选取单词向量维度为 300 去构建文本向量去验证本文所提方法的有效性.

• 问题 1: 数据预处理是否可以应对需求文本存在语义模糊性的问题, 准确识别其隐含的时序语义?

在数据预处理阶段, 对需求文本进行数据清洗以降低噪声, 并使用词性标注来汇集带有时序标签的特征信息, 同时删除信息熵较低的单词. 此外, 通过依存关系分析推断出句子中单词之间的语义关系, 从而提取单词之间的关联性特征信息. 以上提取的特征信息用于突出文本时序逻辑语义特征信息, 并应对自然语言需求文本中语义描述模糊性问题.

为此, 本文使用消融实验测试了数据预处理对实验结果的影响. 图 6 所示的神经网络训练过程, 数据预处理直接影响了模型数据理解能力和性能表现. 本文所设计的预处理模块旨在确保神经网络在训练过程中更有效地学习时序逻辑和语义信息, 表 5 中实验结果表明与不使用预处理相比, 使用预处理步骤能将准确率提升 18.6%–32%. 结果说明数据预处理步骤可以有效突出文本中的时序逻辑语义, 增加模型的性能.



(a) 数据预处理对损失函数的影响 (b) 数据预处理对准确率的影响

图 6 数据预处理对模型结果影响

表 5 启用和禁用数据预处理的准确率对比

模型/是否预处理	100	200	300
FSLNets-TLSA/disable	0.923 809	0.942 857	0.819 047
FSLNets-TLSA/enable	0.723 809	0.714 285	**0.971 428**

• 问题 2: 小样本网络相较于其他神经网络多分类网络是否能在需求样本中存在的数据样本稀疏和不平衡下更准确地识别时序逻辑语义?

本文完成了与多个多分类神经网络模型的对比试验, 实验结果展现了所提网络结构的性能优势. 小样本学习通过从表 2 中的类别随机选择标记样本作为支持集, 并从每类中选取剩余的一部分作为查询集, 支持集与查询集分别用作网络的输入和输出前评估的样本. 支持集和查询集的排列组合可以有效缓解数据样本稀疏和不平衡问题, 以及需求冷启动问题. 在网络构造上, 网络结构由编码器, 归纳模块和关系模块堆叠构成, 用以捕捉文本中隐含的时序逻辑语义信息. 编码器使用 LSTM 以增加对长文本语义的理解能力, 在归纳模块应用动态路由算法, 共享样本特征信息, 提升输出和真实预测结果的耦合系数. 在与多种多分类神经网络模型对比下, 观察到 FSLNets-TLSA 的性能表现如表 6 所示.

根据表 6 实验结果, 本文可以更直观地观察各个模型在不同指标上的表现差异. FSLNets-TLSA 模型在损失函数、精确度、召回率和 $F1$ 值上都优于其他模型. 具体而言, FSLNets-TLSA 模型在损失函数上具有最低值 0.24, 损失函数的值越低, 表示模型对训练数据的拟合越好, 从而反映了本模型在预测时的高准确性和在训练学习过程中的高效性. 相比之下, FastText、CNN、RNN、SANN 和 GPT-4 模型的损失值较高, HAN 模型的损失值是最高的. 其原因为, 对比模型的网络结构不适用于捕捉需求文本中的关键时序和逻辑特征信息, 同时这些模型也未对数

据进行专业的预处理, 导致在网络传递时特征信息损失较大, 无法获得准确分类结果. 在精确度方面, FSLNets-TLSA 模型表现最佳, 达到了 0.96, 这表明其预测结果是可靠. SANN 模型在这方面也表现良好, 而其他模型则相对较低. 在召回率方面, FSLNets-TLSA 同样表现最佳为 0.96, 显示出其在捕捉正类别样本的能力较强. SANN 模型使用 LSTM 模型和自注意力机制, 和本文模型结构存在部分相似性, 其实验结果也较为优异. GPT-4 的损失由 API 返回的 Loss 转化计算得出的近似值, 其依赖于庞大模型和数据支持, 在语义解析中表现优异. 在 $F1$ 值方面, FSLNets-TLSA 模型取得了最佳结果 0.96, $F1$ 得分是精确度和召回率的综合度量, 因此综合来看, 本文模型在损失值、精确度、召回率和 $F1$ 值这 4 个方面均表现出色, 显示出其网络结构在识别时序逻辑语义任务中的优异性能.

表 6 语义分析模型对比

模型	Loss	Precision	Recall	$F1$
FastText[34]	0.609382	0.928826	0.928826	0.928826
CNN[35]	0.916068	0.900446	0.927639	0.915691
RNN[36]	1.65727	0.928826	0.942857	0.928826
SANN[37]	1.25184	0.940828	0.943060	0.941943
HAN[38]	5.76862	0.928826	0.928826	0.928826
GPT-4[39]	0.435971	0.932974	0.942687	0.936806
FSLNets-TLSA	0.240276	**0.965517**	**0.962912**	**0.964213**

• 问题 3: 模型增强模块是否能应对监控语义分析准确率较低的问题?

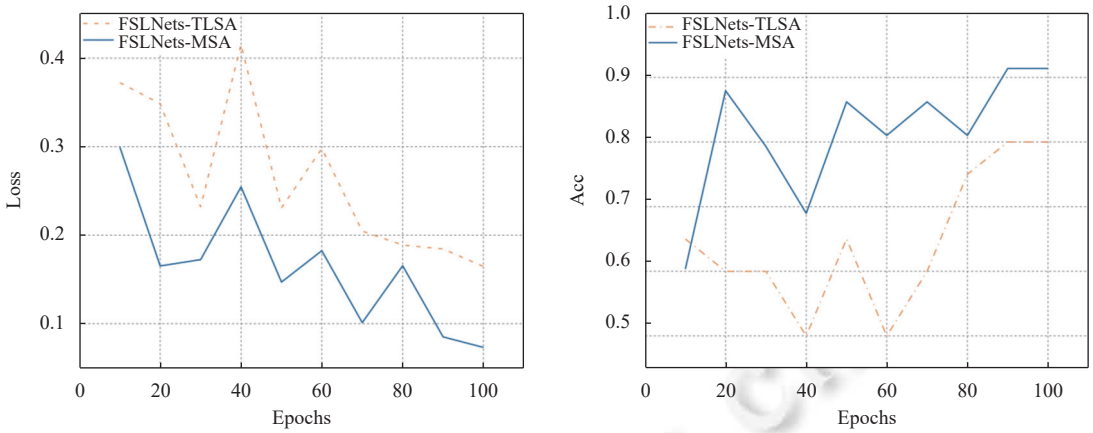
监控语义是 PPTL 在描述多核并行程序进程间交互起到关键作用, 尤其是在处理并发和通信方面. 监控语义通过投影算子和同步通信机制, 不仅支持并发进程以不同的执行速率动态执行, 还能确保在交汇点处进程间的变量共享, 从而保障系统行为的一致性和安全性. 这些特性使监控语义成为形式化验证中关注的重点, 尤其是在验证多核并行程序的安全性质时. 这一点体现在其能够细致地捕捉到进程间的精确交互和同步要求, 是其他形式规约难以覆盖的. 因此, 本文将监控语义与模型增强模块的设计紧密关联, 反映了监控语义在确保并行程序正确性中的核心作用, 也符合提高形式规约质量的必要性的要求, 这对于形式化验证的成功实施是不可或缺的.

以 Java.Io 中一个标注为监控语义的需求文本为具体实例, 解释监控语义具体含义. 在进行 Java 程序的运行时验证时, 本文可通过逻辑表达式来明确操作的序列性要求. Java.Io 需求明确规定要确保程序在写数据库之前必须先建立数据库连接, 且此顺序不能颠倒. 设命题 c 表示程序连接数据库的行为, 命题 w 表示程序写入数据库的行为. PPTL 公式 $\square \neg w;c$ 可用于描述这一性质. 根据本文第 2.4 节监控语义定义, 可对该性质进行解释: 在监控 Java 程序的过程中, 若观察到某个时刻程序正在连接数据库, 并且在此之前未发生写数据库的操作, 则可断定程序在数据库操作步骤上是正确的, 验证结果为 true. 相反, 若观察到程序在未先连接数据库的情况下正在写入数据库, 则表明程序违反了设定的性质, 验证结果为 false. 若在执行过程中既没有连接数据库的操作, 也没有写数据库的操作, 那么无法判断性质是否成立, 此时验证结果为 *inconclusive*.

本文统计了在数据集中标注为监控语义的语句, 极其稀疏仅占 4%, 远小于训练所需的样本规模, 也难以维持与其他时序标签样本的平衡性. 因此为了适应监控语义识别, 本文继续引入并改良了同义词替换和回译方法对监控语义数据增强, 并集成了对抗训练模块, 以增强神经网络模型的有效性和鲁棒性, 解决了模型的过拟合问题. 为了分析本模块添加的两个方法对模型性能的影响, 本文使用消融实验验证其两者性能, 在监控语义数据集中, 对比了添加 EDA 数据增强和 PGD 对抗训练的实验效果, 准确率的结果如表 7 和图 7 所示.

表 7 消融实验准确率对比结果

模型	准确度
FSLNets-TLSA	0.870370
FSLNets-TLSA+EDA	0.907407
FSLNets-TLSA+PGD	0.944444
FSLNets-MSA	**0.956790**



(a) 模型增强对损失函数的影响 (b) 模型增强对准确率的影响

图 7 数据预处理对模型结果影响

在只添加 EDA 数据增强的情况下, 数据集的准确率明显提升了 3.71%, 显示出 EDA 方法在提高模型性能方面的有效性. 相对地仅通过添加 PGD 对抗训练也观察到了准确率的增加, 这表明 PGD 生成的对抗样本也能够有效提升模型的防御能力. 这使得模型能够更好地应对各种输入扰动, 从而增强了模型的鲁棒性和泛化能力. 改进的模型 FSLNets-MSA 有效地结合了这两种技术, 显著提高了分类效果, 在监控语义的识别上展示了卓越的能力. 在复杂的监控语义分析中, 本文同样在 FSLNets-MSA 与其他相似工具做了对比试验, 如表 8 所示.

表 8 语义分析模型对比

模型	Precision	Recall	*F*1
ATAE-LSTM[40]	0.484 568	0.500 000	0.492 163
SVM[41]	0.896 815	0.896 815	0.896 815
BERT[23]	0.718 954	0.884 076	0.776 037
GPT-4[39]	0.895 817	0.905 495	0.899 573
FSLNets-MSA	**0.926 575**	**0.918 301**	**0.922 238**

表 8 中可以清楚地看到不同模型在精确度、召回率和 *F*1 值上的表现. 显然 FSLNets-MSA 模型在这 3 个指标上均表现最好. FSLNets-MSA 模型在精确度方面相比于 ATAE-LSTM[40]、SVM[41]、BERT[23]和 GPT-4[39]显示出更高的数值, 说明该模型在正确预测正类别上更为有效. FSLNets-MSA 的召回率为 0.918, 此数值高于其他模型, 表明其在捕捉监控语义方面的出色表现. SVM 优势在于其泛化和避免过拟合能力, 尤其是在训练样本较少的情况下, 该模型的核函数的选择和正则化参数的设置对监控语义识别也适用, 模型取得了不错的效果. ATAE-LSTM 虽然也使用 LSTM 架构, 但是其过度关注每个词的权重, 导致数据集中大量的时序逻辑不相关的词受到过多的关注, 从而造成分类结果不理想. BERT 训练主要是基于掩码语言模型, 完成下一个句子的预测任务, 主要采用模型预训练加微调的策略, 然而需求文本中语义模糊的句子和数据不平衡削弱了微调效果, 因而未能获得理想的语义分析效果. GPT-4 尽管具有丰富的先验知识和复杂的模型构造, 但是当它缺乏有效的样本提示和模型参数微调, 很难获得最佳性能. 最终 FSLNets-MSA 模型的 *F*1 得分为 0.922, 超过了其他模型, 进一步证明了本论文所提方法在综合性能上的优越性. 总体而言, 设置合理的参数配置词向量维度为 300, 可以保证本文模型逐渐收敛至一个较好的性能, 实验结果也表明, 本文模型可以有效地捕获需求文本中的时序逻辑语义, 解决目前时序逻辑语义研究 3 个的主要难点.

• 问题 4: 隐含语义识别是否对形式化规约生成准确率产生影响?

隐含语义识别能够辅助形式化规约自动生成工作, 通过神经网络捕捉隐含的时序特征信息, 将需求划分为特

定的时序标签, 确保生成的时序逻辑公式能够准确表达需求中的时序、依赖关系和复杂行为, 从而减少需求描述中的歧义与误解, 整体提升形式化规约生成的准确率与有效性. 本文以 ICON 数据集中隐含语义为例, 如表 9 所示, 说明需求文本中难以识别的隐含语义的分类, 及其对于形式化规约生成准确率的影响.

表 9　隐含语义识别对形式化规约生成影响

数据集	总计	前提条件	并行或允许的操作	警告或重要提示
Amazon S3 REST API	14	2	4	8
PayPal Payment REST API	20	2	7	11
Java.Io	47	12	6	29
形式化规约准确率 (%)	↑3.05	↑0.60	↑0.64	↑1.81

(1) 前提条件. 需求描述某个操作之前必须满足的条件或必须先完成的步骤. 如数据集中“However, for copying an object greater than 5 GB, you must use the multipart upload API”.

(2) 并行或允许的操作. 需求描述了可以与其他操作同时或在不同时间段内进行的操作. 如“However, bucket owners can grant other users permission to delete the website configuration by writing a bucket policy granting them the S3 permission”.

(3) 警告或重要提示. 需求强调了某些操作的关键步骤或后续步骤的要求. 如“Amazon S3 returns a unique identifier, the upload ID, that you must include in your upload part request”.

隐含语义相较于显式语义, 在需求数据中体现为语法描述自由且占比较少, 很难制定的专门的语法规则去定义和识别, 导致 ICON 工具中无法直接将其自动转化为逻辑公式. 本文所提技术可以解决隐含语义在形式化规约生成过程中无法识别的问题, 提升形式化规约生成的准确率. 同时通过将需求文本标注上清晰的时序标签分类, 使得规约更易于理解和维护, 这在实际形式化验证中也能够减少错误和偏差, 提升形式化规约的可用性.

• 问题 5: 模型是否具备跨领域软件需求文档的泛化能力?

在分析跨领域软件需求文本的时序逻辑语义时, 模型的泛化能力至关重要. 尽管本模型在特定任务上表现优异, 但其在不同领域需求文档中的适应性仍需进一步评估. 为此, 本实验通过引入多个不同领域的软件需求数据集, 评估模型在跨领域任务中的泛化性能, 以确保方法的可复现性和科学严谨性. 该模型实际应用于航天太阳搜索控制软件子系统项目研究[42], 并在实际业务场景中取得了良好的效果, 同时额外引入 ARSC (Amazon review sentiment classification) 软件产品功能需求文档、自动驾驶需求文档进行了实验评估, 在词向量维度和 Epoch 均为 300 情况下, 采用领域自适应微调技术将新需求文档带入微调模型参数, 分析模型不同领域需求文档的性能表现. 以下是软件需求文件介绍.

(1) 在太阳搜索控制需求文档 16 个 IP 的软件需求, 其均为可形式化验证的安全需求描述.

(2) ARSC 开源数据集中 Software.t2 产品类别中, 此数据集为软件开发者提供了多种类型软件的需求, 包括描述软件功能、兼容性以及数据处理. 开发者可以深入了解用户对软件功能的实际需求和期望, 从而更精确地定义功能需求.

(3) 自动驾驶的软件数据集, 描述了自动驾驶系统与传感器的交互数据、系统响应和安全机动的执行指令, 涵盖了自动驾驶在正常运行和故障模式下的安全需求描述.

表 10 显示了模型在不同领域需求文本下的良好的泛化能力和适用性. 在对 3 个软件需求文档数据集的模型性能评估中, 模型在描述规范和技术性较强的文档, 如太阳搜索控制需求文档, 表现出高精度和准确率, 其准确率达到 1.0, 表明模型对该领域需求的语义解析具有适用性, 能够精准地处理规范化安全需求描述. 而在处理广泛软件功能的 ARSC 数据集时, 模型准确率为 0.79 有所下降, 这与数据集中需求文本的多样性和技术描述的不一致性有关. 对于自动驾驶需求数据集, 模型展现出较高的适应性, 语义解析准确率达到 0.94, 表明模型在处理自动驾驶领域需求时仍能保持较高的准确性. 这些结果验证了模型在解析需求时序逻辑的有效性, 同时也表明当数据集内容包含多样且文本结构差异较大需求时, 模型的性能可能会有所下降. 实验代码及数据已发布在 https://github.com/ChunyiLi322/TLSA.

表 10 模型在跨领域软件需求文档上的准确性评估

数据集	需求文本数量	Loss	Precision	Accuracy
太阳搜索控制需求	79	0.006 849	0.962 962	1.0
ARSC	795	0.201 903	0.912 500	0.796 296
自动驾驶需求	74	0.054 785	0.791 666	0.944 444

5 总 结

本文介绍了一种从自然语言需求文本中识别时序逻辑语义的方法 FSLNets-TLSA, 该方法对需求文本进行预处理, 包括 POS 标记、数据降噪、依存分析以及文本向量化等步骤, 并使用小样本网络识别文本隐含的时序逻辑语义, 同时为了应对更复杂的监控语义, 使用了 EDA 数据增强和 PGD 对抗训练两个方法, 提高了对复杂监控语义的识别准确度. 在后续的工作当中, 将考虑将本文方法与现有的转换工具自动结合, 有望有效地处理自然语言中的隐含时序语义, 从而改进整个自然语言转换为形式逻辑的流程和提高生成形式化规约的质量.

References:

[1] Pnueli A. The temporal logic of programs. In: Proc. of the 18th Annual Symp. on Foundations of Computer Science. Providence: IEEE, 1977. 46–57. [doi: 10.1109/SFCS.1977.32]

[2] Clarke EM, Emerson EA. Design and synthesis of synchronization skeletons using branching time temporal logic. In: Logic of Programs. Yorktown Heights: Springer, 1982. 52–71. [doi: 10.1007/BFB0025774]

[3] Duan ZH, Tian C, Zhang L. A decision procedure for propositional projection temporal logic with infinite models. Acta Informatica, 2008, 45(1): 43–78. [doi: 10.1007/S00236-007-0062-Z]

[4] Chen M, Tam Q, Livingston SC, Pavone M. Signal temporal logic meets reachability: Connections and applications. In: Proc. of the 13th Workshop on the Algorithmic Foundations of Robotics. Cham: Springer, 2020. 581–601. [doi: 10.1007/978-3-030-44051-0_34]

[5] Tolmach P, Li Y, Lin SW, Liu Y, Li ZX. A survey of smart contract formal specification and verification. ACM Computing Surveys, 2022, 54(7): 148. [doi: 10.1145/3464421]

[6] Niang M, Riera B, Philippot A, Zaytoon J, Gellot F, Coupat R. A methodology for automatic generation, formal verification and implementation of safe PLC programs for power supply equipment of the electric lines of railway control systems. Computers in Industry, 2020, 123: 103328. [doi: 10.1016/J.COMPIND.2020.103328]

[7] Wang J, Zhan NJ, Feng XY, Liu ZM. Overview of formal methods. Ruan Jian Xue Bao/Journal of Software, 2019, 30(1): 33–61 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5652.htm [doi: 10.13328/j.cnki.jos.005652]

[8] Bonnah E, Hoque KA. Runtime monitoring of time window temporal logic. IEEE Robotics and Automation Letters, 2022, 7(3): 5888–5895. [doi: 10.1109/lra.2022.3160592]

[9] Srinivasan M, Coogan S. Control of mobile robots using barrier functions under temporal logic specifications. IEEE Trans. on Robotics, 2021, 37(2): 363–374. [doi: 10.1109/TRO.2020.3031254]

[10] Zhou XY, Yang TG, Zou YY, Li SY, Fang H. Multiple subformulae cooperative control for multiagent systems under conflicting signal temporal logic tasks. IEEE Trans. on Industrial Electronics, 2023, 70(9): 9357–9367. [doi: 10.1109/TIE.2022.3215812]

[11] Bruel JM, Ebersold S, Galinier F, Mazzara M, Naumchev A, Meyer B. The role of formalism in system requirements. ACM Computing Surveys, 2022, 54(5): 93. [doi: 10.1145/3448975]

[12] Zhang N, Yang MF, Gu B, Duan ZH, Tian C. Verifying safety critical task scheduling systems in PPTL axiom system. Journal of Combinatorial Optimization, 2016, 31(2): 577–603. [doi: 10.1007/S10878-014-9776-3]

[13] Wang XB, Guo WX, Zhao L, Shu XF. Runtime verification method for social network security based on source code instrumentation. In: Proc. of the 8th Int'l Workshop on Structured Object-oriented Formal Language and Method. Gold Coast: Springer, 2018. 55–70. [doi: 10.1007/978-3-030-13651-2_4]

[14] Zhang N, Wang M, Duan ZH, Tian C. Verifying properties of MapReduce-based big data processing. IEEE Trans. on Reliability, 2022, 71(1): 321–338. [doi: 10.1109/TR.2020.2999441]

[15] Pakonen A, Pang C, Buzhinsky I, Vyatkin V. User-friendly formal specification languages—Conclusions drawn from industrial experience on model checking. In: Proc. of the 21st IEEE IEEE Int'l Conf. on Emerging Technologies and Factory Automation. Berlin: IEEE, 2016. 1–8. [doi: 10.1109/ETFA.2016.7733717]

[16] Zhang SY, Zhai J, Bu L, Chen MS, Wang LZ, Li XD. Automated generation of LTL specifications for smart home IoT using natural language. In: Proc. of the 2020 Design, Automation & Test in Europe Conf. & Exhibition (DATE). Grenoble: IEEE, 2020. 622–625. [doi: 10.23919/DATE48585.2020.9116374]

[17] Pandita R, Taneja K, Williams L, Tung T. ICON: Inferring temporal constraints from natural language API descriptions. In: Proc. of the 2016 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Raleigh: IEEE, 2016. 378–388. [doi: 10.1109/ICSME.2016.59]

[18] He J, Bartocci E, Ničković D, Isakovic H, Grosu R. DeepSTL: From English requirements to signal temporal logic. In: Proc. of the 44th Int'l Conf. on Software Engineering. Pittsburgh: ACM, 2022. 610–622. [doi: 10.1145/3510003.3510171]

[19] Zhong H, Zhang L, Xie T, Mei H. Inferring resource specifications from natural language API documentation. In: Proc. of the 2009 IEEE/ACM Int'l Conf. on Automated Software Engineering. Auckland: IEEE, 2009. 307–318. [doi: 10.1109/ASE.2009.94]

[20] Fuggitti F, Chakraborti T. NL2LTL—A Python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas. In: Proc. of the 37th AAAI Conf. on Artificial Intelligence. Washington: AAAI Press, 2023. 16428–16430. [doi: 10.1609/AAAI.V37I13.27068]

[21] Wang XB, Liu DM, Zhao L, Xue YN. Runtime verification monitor construction for three-valued PPTL. In: Proc. of the 6th Int'l Workshop on Structured Object-oriented Formal Language and Method. Tokyo: Springer, 2016. 144–159. [doi: 10.1007/978-3-319-57708-1_9]

[22] Wu Q. Research on temporal logic semantic analysis of natural language requirements [MS. Thesis]. Xi'an: Xidian University, 2024 (in Chinese with English abstract).

[23] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional Transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis: ACL, 2019. 4171–4186. [doi: 10.18653/V1/N19-1423]

[24] Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding by generative pre-training. 2018. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[25] Yang ZL, Dai ZH, Yang YM, Carbonell J, Salakhutdinov R, Le QV. XLNet: Generalized autoregressive pretraining for language understanding. arXiv:1906.08237, 2019.

[26] Blasi A, Gorla A, Ernst MD, Pezzè M. Call me maybe: Using NLP to automatically generate unit test cases respecting temporal constraints. In: Proc. of the 37th IEEE/ACM Int'l Conf. on Automated Software Engineering. Rochester: ACM, 2022. 19. [doi: 10.1145/3551349.3556961]

[27] Pandita R, Xiao XS, Yang W, Enck W, Xie T. WHYPER: Towards automating risk assessment of mobile applications. In: Proc. of the 22nd USENIX Conf. on Security. Washington: USENIX Association, 2013. 527–542.

[28] Chen YB, Liu Z, Xu HJ, Darrell T, Wang XL. Meta-baseline: Exploring simple meta-learning for few-shot learning. In: Proc. of the 2021 IEEE/CVF Int'l Conf. on Computer Vision. Montreal: IEEE, 2021. 9042–9051. [doi: 10.1109/ICCV48922.2021.00893]

[29] Bushara AR, Kumar RSV, Kumar SS. Classification of benign and malignancy in lung cancer using capsule networks with dynamic routing algorithm on computed tomography images. Journal of Artificial Intelligence and Technology, 2024, 4(1): 40–48. [doi: 10.37965/jait.2023.0218]

[30] Socher R, Chen DQ, Manning CD, Ng AY. Reasoning with neural tensor networks for knowledge base completion. In: Proc. of the 27th Int'l Conf. on Neural Information Processing Systems. Lake Tahoe: Curran Associates Inc., 2013. 926–934.

[31] Ahn J, Chang K, Choi KM, Kim T, Park H. DTOC-P: Deep-learning-driven timing optimization using commercial EDA tool with practicality enhancement. IEEE Trans. on Computer-aided Design of Integrated Circuits and Systems, 2024, 43(8): 2493–2506. [doi: 10.1109/TCAD.2024.3370110]

[32] Wei J, Zou K. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: ACL, 2019. 6382–6388. [doi: 10.18653/V1/D19-1670]

[33] He LX, Wang Z, Yang SS, Liu T, Huang YM. Generalizing projected gradient descent for deep-learning-aided massive MIMO detection. IEEE Trans. on Wireless Communications, 2024, 23(3): 1827–1839. [doi: 10.1109/TWC.2023.3292124]

[34] Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification. In: Proc. of the 15th Conf. on the European Chapter of the Association for Computational Linguistics. Valencia: ACL, 2017. 427–431. [doi: 10.18653/V1/E17-2068]

[35] Zhang Y, Wallace B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In: Proc. of the 8th Int'l Joint Conf. on Natural Language Processing. Taipei: Asian Federation of Natural Language Processing, 2017. 253–263. [doi: 10.18653/v1/I17-1026]

[36] Liu PF, Qiu XP, Huang XJ. Recurrent neural network for text classification with multi-task learning. arXiv:1605.05101, 2016.

[37] Lin ZH, Feng MW, dos Santos CN, Yu M, Xiang B, Zhou BW, Bengio Y. A structured self-attentive sentence embedding. arXiv:1703.03130, 2017.

[38] Yang ZC, Yang DY, Dyer C, He XD, Smola A, Hovy E. Hierarchical attention networks for document classification. In: Proc. of the 2016 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego: ACL, 2016. 1480–1489. [doi: 10.18653/V1/N16-1174]

[39] Achiam J, Adler S, Agarwal S, *et al*. GPT-4 technical report. arXiv:2303.08774, 2023.

[40] Wang YQ, Huang ML, Zhu XY, Zhao L. Attention-based LSTM for aspect-level sentiment classification. In: Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing. Austin: ACL, 2016. 606–615. [doi: 10.18653/V1/D16-1058]

[41] Vo DT, Zhang Y. Target-dependent Twitter sentiment classification with rich automatic features. In: Proc. of the 24th Int'l Conf. on Artificial Intelligence. Buenos Aires: AAAI Press, 2015. 1347–1353.

[42] Chen XH, Liu SB, Jin Z. Survey on requirements description of embedded system. Ruan Jian Xue Bao/Journal of Software, 2025, 36(1): 27–46 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7157.htm [doi: 10.13328/j.cnki.jos.007157]

附中文参考文献:

[7] 王戟, 詹乃军, 冯新宇, 刘志明. 形式化方法概貌. 软件学报, 2019, 30(1): 33–61. http://www.jos.org.cn/1000-9825/5652.htm [doi: 10.13328/j.cnki.jos.005652]

[22] 武强. 自然语言需求的时序逻辑语义分析研究 [硕士学位论文]. 西安: 西安电子科技大学, 2024.

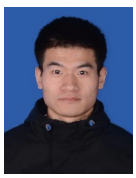
[42] 陈小红, 刘少彬, 金芝. 嵌入式系统的需求描述综述. 软件学报, 2025, 36(1): 27–46. http://www.jos.org.cn/1000-9825/7157.htm [doi: 10.13328/j.cnki.jos.007157]

李春奕(1996－), 女, 博士生, 主要研究领域为形式化方法, 自然语言处理.

王小兵(1979－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为高可信软件, 形式化验证.

马智(1994－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为操作系统, 形式化验证, 嵌入式软件.

赵亮(1984－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为神经网络建模与验证, 形式化验证, 时序逻辑程序设计.



武强(1998－), 男, 硕士生, 主要研究领域为软件工程, 形式化方法.