计算机科学
COMPUTER SCIENCE

面向业务的资源按需解析模型构建研究

刘耀，秦迅，刘天吉

引用本文

刘耀，秦迅，刘天吉. 面向业务的资源按需解析模型构建研究[J]. 计算机科学，2024，51(10)：178-186.

LIU Yao, QIN Xun, LIU Tianji. Study on Building Business-oriented Resource On-demand Resolution Model [J]. Computer Science, 2024, 51(10): 178-186.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230800191

面向业务的资源按需解析模型构建研究

刘 耀[1] 秦 迅[2] 刘天吉[2]
1 中国科学技术信息研究所 北京 100038
2 北京大学软件与微电子学院 北京 102600

摘 要 针对在项目开发过程中新需求来临时,需要对自然语言处理工具和资源解析插件进行重新需求分析、重复开发等问题,提出了一套面向业务的资源按需解析方案。首先,提出了一种从需求到代码的资源按需解析方法,针对需求文本本身进行需求概念标引模型的构建。构建的需求概念标引模型的准确率、召回率、F1值等指标均高于其他分类模型。然后,针对需求文本与代码的关联,建立从需求文本到代码库类别的映射机制。对于模型的映射结果,使用前K准确率(percision@K)作为评价指标,最终准确率达到60%,具有一定的实用价值。综上所述,探索了一套具有需求解析能力、实现需求与代码关联的资源按需解析关键技术,并贯穿需求文本分类、需求代码库分类、代码库检索到插件生成的整个流程,形成了完整的"需求-代码-插件-解析"的业务闭环,通过实验验证了所提方法对于资源按需解析的有效性,为业务需求分析与软件复用提供了思路,与现有用于业务需求的解析和代码生成的大语言模型相比,所提方法聚焦于具体业务领域内的含有业务特点的插件代码复用全流程的实现。

关键词:自然语言处理;需求模型;代码复用;文本解析;代码分类;代码检索
中图分类号 TP391

Study on Building Business-oriented Resource On-demand Resolution Model

LIU Yao[1],QIN Xun[2] and LIU Tianji[2]
1 Engineering Center,Institute of Scientific and Technical Information of China,Beijing 100038,China
2 School of Software and Microelectronics,Peking University,Beijing 102600,China

Abstract To address the issue of re-analyzing and repeating development of natural language processing tools and resource analysis plugins when new requirements arise during project development,this paper proposes a business-oriented on-demand resource analysis solution. Firstly,a demand-driven resource analysis method from requirement to code is proposed,focusing on the construction of a demand concept indexing model for the requirement text itself. The constructed demand concept indexing model outperforms other classification models in terms of accuracy,recall,and F1 score. Secondly,this paper establishes a mapping mechanism from requirement text to code library categories based on the correlation between requirement text and code. For the mapping results,the precison@K is used as an evaluation metric,with an ultimate accuracy rate of 60%,demonstrating a certain practical value. In summary,this paper explores a set of key technologies for on-demand resource analysis with demand parsing capabilities and implements the correlation between requirements and code,covering the entire process from requirement text classification,code library classification,code library retrieval to plugin generation. The proposed method forms a complete business loop of "requirement-code-plugin-analysis" and experimentally verifies to be effective for on-demand resource analysis. Compared to existing large language models for business requirement analysis and code generation,this method focuses on the implementation of the full process of plugin code reuse within specific business domains,containing business characteristics.

Keywords Natural language processing,Requirements model,Code reuse,Text parsing,Code categorization,Code retrieval

1 引言

互联网的发展催生了各种各样的业务需求,使信息的呈现形式更为丰富多样,不同的业务需求及软件开发任务为信息资源的处理与交换带来了挑战。在项目开发过程中,根据不同的业务需求,开发人员针对性地开发插件,使计算机能够理解、读取并利用这些信息资源,将半结构化、无结构或结构化的信息资源转化为符合业务需要的结构化信息资源,实现对目标资源的结构化解析或信息提取,并按照特定的方式使用、提供服务。

随着业务需求的增多,研发所面临的工作量逐渐增大,且存在重复劳动。本文基于项目中积累的大量解析插件,能够

到稿日期:2023-08-30 返修日期:2024-01-22
基金项目:国家社会科学基金(21BTQ011)
This work was supported by the National Social Science Foundation of China(21BTQ011).
通信作者:刘耀(liuy@istic.ac.cn)

对插件所能满足的业务需求进行针对性分析，以期在新的业务需求来临时，从系统自动生成能满足业务需求的解析插件，实现资源的按需处理，提高工作效率，减少重复劳动。譬如针对“对政策文本进行人名和地名识别”这一业务需求文本，现有大模型技术无法快速针对具体业务特点进行相应的代码生成工作，本文提出的方法建立在团队之前的业务处理基础上，通过对该业务需求文本进行解析来获取业务特征，并将业务特征映射到团队积累的业务资源解析插件，在处理信息资源即政策文本时调用相应的资源处理工具插件，完成对信息资源的解析处理，最终生成业务需求所表述的“人名”“地名”等目标资源。

在此背景下，本文探索了一套面向业务的资源按需解析方法，基于现有资源类型，通过处理大量的业务需求及案例生成实验，构建按需解析模型，实现对绝大部分业务需求的覆盖。基于上述模型，研发相关技术或使用相关算法，生成满足业务需求的解析插件，实现面向业务的资源按需解析，最后以具体业务验证该模型的有效性，为瞬时决策及快速情报系统提供资源与技术支持。

针对项目中的真实业务需求，本文以平台中的资源解析流程为基础，提出了一种从需求到代码的资源按需解析方法，连通了从业务需求、算法代码分类到插件代码生成的整个流程，形成完整的“需求-代码-插件-解析”的业务闭环，能够应用于实际业务。本文的具体工作如下：

1）提供业务需求文本解析技术。基于需求文本的语言特点，构建需求文本资源库；结合需求短文本的行文结构，构建需求文本的概念模型，从而对需求文本进行概念标引，对所属的概念结构类型进行解析，并映射到相应的需求处理类型。

2）探索了从业务需求到资源解析的可行性。本文以需求文本为入口，通过构建需求文本概念结构模型、算法代码分类模型、算法代码检索模型，提供了从需求文本到插件代码的全流程实现方法，快速实现资源按需解析，提升软件项目的开发效率。

2　相关工作

2.1　命名实体识别

命名实体识别任务对文本进行实体抽取来获取概念，作为构建本体或知识图谱的基础。现阶段主要使用基于有监督的深度学习方法，包括在神经网络结构上加入注意力机制，目前的主流方法为基于序列标注并采取概率约束的 Bi-LSTM＋CRF 模型[1]以及预训练的大规模语言模型 BERT 模型[2]等。LSTM 具有较强的序列特征提取能力，能够获得丰富的编码信息，将输入转化为标注序列，结合 CRF 通过观察序列，预测隐藏的状态序列。采用深度学习方法是为了避免构建大量人工特征，深度学习方法能够构建复杂的网络，直接进行端到端的训练。目前通常使用基于字序列标注的深度学习模型，而该类模型也有一定的局限性，即只能应用在有限的领域和有限的实体类型中，因此本文结合需求文本特征，使用文本分类的方法提取文本中的概念。

2.2　文本分类

在自然语言处理任务中，首先要将文本转变成让计算机更容易处理的形式，先对自然语言文本进行预处理，即对文本进行特征表示。目前通常采用有监督的深度学习方法，深度学习方法通过学习一组非线性变换将特征集成到输出中，直接进行模型的拟合，利用神经网络的结构自动学习到文本的特征并进行表示，能够端到端地解决问题，因此其经常被应用在文本分类任务中并有较好的表现。经典的文本分类模型主要基于卷积神经网络（CNN）[3]、循环神经网络（RNN）[4]模型及其变种。与 CNN 模型相比，RNN 模型假设事物的发展按照时间顺序展开，在自然语言处理任务中，即假设每个时间步的输出取决于当前的单词及其之前的单词，将词进行词嵌入后送入神经元，通过在时间序列上共享权重矩阵 $\boldsymbol{W}$，保留了上一时间步下神经元的输出，最后通过 softmax 计算得到不同类别标签的概率分布。因此，RNN 能够捕捉到一定长度内词语间的依赖关系，非常适合于时间序列问题，因此较多地被应用于自然语言处理任务。CNN 模型是仿照生物的视觉机制构建的，核心是使用多个不同的卷积核，同时在不同的空间位置上共享卷积核参数来捕获局部特征，并在池化层进行特征选择和过滤，最终将特征输出为文本的类别标签。Kim[5]提出了 CNN 文本分类模型，在句子级别的分类中取得了较好的效果，也证明了预训练词向量对预测结果的重要性。Shi 等[6]提出了循环卷积神经网络，其融合了 RNN 的结构和 CNN 的池化层的优势，能更准确地表示文本的语义。然而，前述 RNN 的循环机制过于简单，梯度在反向传播时的连乘导致了梯度消失和梯度爆炸问题。基于 RNN 的一些变种模型一定程度上缓解了这个问题，如 Hochreiter 等[7]提出了长短时记忆神经网络（LSTM），解决了 RNN 容易出现梯度消失的问题；Gers 等[8]引入了遗忘门机制，遗忘掉不重要信息；Graves 等[9]在已有工作基础上提出了双向的长短时记忆神经网络，成为当前应用最广泛的 LSTM 模型；Chung 等[10]提出了 GRU 模型，摆脱了单元状态，简化了 LSTM 架构，进一步解决了梯度消失和梯度爆炸的问题。Peters 等[11]提出先基于大规模预料预训练双向 LSTM 模型，学习基于上下文的语义表示，再根据下游任务进行微调，能够提高原模型的表现。随后基于 Transformer 的 BERT 和 GPT 模型[12]都采用了预训练＋微调的方式处理各项任务。然而，深度学习需要大量的标注样本，在训练数据不足时，直接利用预训练模型中现有的参数进行预测，无法充分利用模型的知识，训练结果不够高效。本文使用大规模预训练模型中的 MLM 模型作为生成模型，将概念标引问题转化为文本生成问题，根据任务描述构建相应的完形填空模板，进行小样本下的监督学习，进而将概念分类任务转化为完形填空任务。

2.3　代码检索

现有的代码检索研究所涉及的技术分为基于信息检索和基于神经网络两大类。

基于信息检索的技术代码检索假定源代码语料库由自然语言文本构成，依靠查询和源代码在词级别上的相似度进行检索，本质上是对查询和源代码进行相关度建模。代码中的函数、类、变量名等通常为单词缩写或相互组合，因此查询和代码中语义更相似的词汇能够代表两者的相关程度。Lu 等[13]利用词间关系，为查询语句完成基于关键词的代码

匹配。Peters 等[14]结合关键字匹配和 PageRank 方法查询代码中的函数代码片；Lv 等[15]提出了 API 匹配的代码推荐工具 CodeHow；Rahman 等[16]针对检索时代码语义信息不足的问题，使用程序问答网站的内容对查询进行扩展。

随着深度学习技术的发展与源代码数量的积累，许多研究提出采用神经网络模型，对所有源代码和查询进行联合建模，并映射到同一个向量空间中，为每一条输入的文本或代码进行序列标注，把语义上相关的代码和查询映射到相近的向量空间中。Github 等公司的 Husain 等[17]使用序列标注模型，将开源软件中的函数代码与对应文档中的自然语言进行匹配；Gu 等[18]提出了基于 RNN 的 Code-Description Embedding Neural Network(CODEnn)模型，创建代码和文档的语义表示。CODEnn 将 JAVA 源代码中的方法名、API、方法中的字符 token 和文档描述嵌入到同一高维向量空间中，以获得相应的语义表示。将代码与文本共同嵌入，对于用户的查询，CODEnn 将查询转化为向量，使用余弦相似度进行计算，返回语义最相近的代码片段。

由于源代码从编译和解释的角度来看具有高度结构化的特征，因此也有相关研究引入了代码的结构信息。Yin 等[19]提出同时使用抽象语法树和控制流图等代码的结构信息来表示代码的语义，在代码任务中证明了引入结构信息的有效性。Zhang 等[20]基于 AST 与双向 RNN 的神经网络(ASTNN)的源代码表示模型，把代码的整个抽象语法树分割成多个小型语法树，提取代码片段的结构特征，生成代码片段的向量表示，并在代码分类以及克隆检测任务上验证了模型的有效性。Wan 等[21]提出使用多模态注意力网络对代码的非结构化和结构化特征进行表述，即采用 LSTM 对分词后的代码序列进行表示，使用 Tree-LSTM 对代码的 AST 进行表示，以及使用门控图神经网络(Gated Graph Neural Network，GGNN)对代码的控制流图进行表示，并通过代码片段检索实验验证了模型的有效性。同时，研究结论还表明 CFG 中的部分语句可能对代码的执行结果没有直接贡献，因此可能会使代码表示产生一定的偏差。本文基于融合需求语义的代码关键词表示技术，将代码库的所有代码进行向量化表示，从结构层面对代码文档进行检索，实现面向需求文本的算法获取。

3 业务按需解析模型

本文针对不同业务需求，实现资源的解析，主要研究两方面内容：1)需求文本解析，根据解析结果确定需求文本的概念关系结构，实现需求处理类型的分类；2)插件代码生成，基于对算法代码语义的理解，构建算法代码的分类模型，针对不同的需求处理类型，结合需求文本确定算法代码类别，然后针对不同类别下的算法代码实现算法代码检索，完成从需求文本到需求处理类型、需求文本到算法代码类别，最终到具体算法代码的匹配机制，最终实现插件代码的按需生成，实现资源按需解析。

3.1 需求文本概念解析模型

本节通过对需求文本的语言特点进行分析，提出了需求文本概念模型，并提出了相应的需求文本概念解析方法。本文在对需求文本资源库构建完毕的基础上，针对所提出的方法，构建了相应的预训练语言模型、分词模型、概念标引模型，以实现对需求文本的概念解析，为后续需求处理类型的判定提供了基础。本节所构建的需求文本解析流程如图 1 所示。

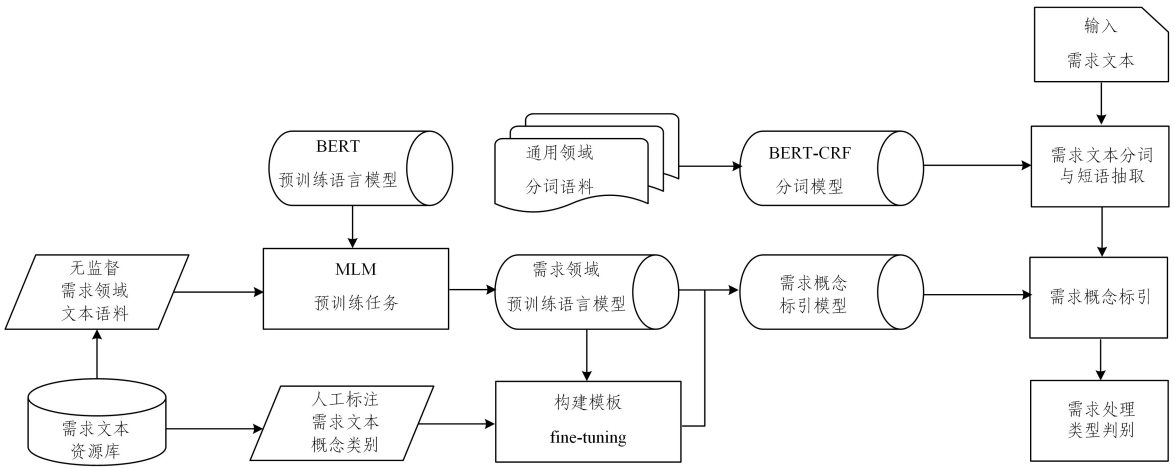

图 1 需求文本解析的流程

Fig. 1 Requirements text parsing flowchart

3.1.1 概念模型构建

本文基于项目需求描述文档、概要设计书、详细设计书、插件代码的描述等相关文档对概念进行建模，最终总结出的业务需求文本概念类型如表 1 所列。

本文针对平台所有业务，选择从文本的特点出发，以文本处理的颗粒度对业务需求进行划分，按照词汇构成句子、句子构成篇章的文本特点，将需求按照处理类型划分为词汇级处理类型、句子级处理类型、篇章级处理类型，并认为这种对需求类型的划分方式能够覆盖大多数文本处理的需求。本文通过定义每种需求类型的概念结构特征，来确定业务所属的需求处理类型。本文假设，针对本文研究的只处理单一需求的无结构需求文本，都具有“需求-处理”式结构，即在需求文本中至少包含一个“需求”类概念与一个“处理”类概念。在本文构建的需求概念结构中，针对具体业务，将“需求”类概念细分为“词汇”“句子”和“篇章”类概念，构建相应的需求文本概念结构模型，如表 2 所列。

表 1　业务需求文本概念模型

Table 1　Conceptual model of business requirements text

概念分类		说明	示例
需求	词汇类	表达词汇类需求的概念	词、关键词、地名、人名
	句子类	表达句子类需求的概念	观点句、结论句、主题句
	篇章类	表达篇章类需求的概念	摘要、文档、综述
处理		进行处理所需要的操作	提取、生成、切分、解析、构建
业务领域		业务主要领域	科技、中医、地理、新闻
技术方法		涉及的算法或工具	textrank、最大熵模型
来源		业务需求的来源描述	数据、文本、内容、资源

表 2　需求文本概念结构模型

Table 2　Conceptual structural model of requirements text

需求处理类型	概念结构特征	判断依据	示例
词汇类处理	“词汇类-处理”式结构	词汇类、处理类概念共现，构建“词汇类处理”的概念关系	“人名-识别”
句子类处理	“句子类-处理”式结构	句子类、处理类概念共现，构建“句子类处理”的概念关系	“主题句-提取”
篇章类处理	“篇章类-处理”式结构	篇章类、处理类概念共现，构建“篇章类处理”的概念关系	“摘要-生成”

本文采用基于 MLM(Masked Language Model)的语言模型，将大规模预训练模型中的 MLM 模型作为生成模型。MLM 作为 BERT 的一个预训练任务，随机 Mask 掉文本中的字词，使用模型预测被 Mask 掉的字词，这种训练方式获得的模型能够更好地利用预训练模型中的知识。基于 text-to-text 的思想[22]，本文将需求的概念分类任务视为 text-to-text 的文本生成问题，进一步结合 Pattern-Exploiting Training[23] 的思想，根据任务描述构建相应的完形填空模板，进行小样本下的监督学习，将概念分类任务转化为完形填空任务。

本文将概念标引问题转化为文本生成问题，对于完整句子 $S=\{s_1,s_2,\cdots,s_n\}$，n 为整句中子句或词语的数量，构建模板 $P=\{p_1,p,\cdots,p_n\}$，模板一般为 $P=S+s_n+cloze$，即整句+待标引短句/词语+完形填空短句。在 *cloze* 中，将概念类别所在的字符替换为[MASK]，被遮罩的字符即为所预测的标签。对于概念标引任务，预测标签概率的方法为：定义一个映射函数 v，将可能的字符集映射到标签 l，即 $v(l)$。对于输入的短句或词语 s_n，其标签为 l 的分数为：

$$score_p(l|s_n)=M(v(l)|p_n) \tag{1}$$

其中，l 为标签 label；$v(l)$ 为 label 的映射，即 *verblizer*；M 为模型；p_n 为构建的模板；$score_p$ 为 label l 的分数，即模型对 mask 位置预测的 logits。随后使用 softmax 函数获得各标签的概率分布，得到最大概率的 token。

$$q_p(l|s_n)=\frac{e^{score_p(l|s_n)}}{\sum_{l'}e^{score_p(l'|s_n)}} \tag{2}$$

3.1.2　语料准备

本文从多个系统的需求文档、详细设计说明书、概要设计说明书，以及针对多种自然语言处理任务的论文摘要中，分别选取了不同类型的句子进行概念标引，将句子中的词语或短语视为最短语义单元，并标注概念的分类，标注示例如下：

例 1　识别(选自《政策平台需求概设文档》)

使用 LSTM-CRF 模型识别中医临床症状术语。

例 2　关键词(选自论文《一种基于词汇链的关键词抽取方法》)

通过计算词义相似度首先构建词汇链，然后结合词频与区域特征进行关键词选择。

例 3　汽车新闻文本(选自论文《基于新闻文本的关键词提取》)

以汽车新闻文本作为研究数据，通过 TextRank 图模型和 Word2Vec 相结合的方法，提取汽车新闻文本的关键词。

根据本文构建的需求文本概念模型，在例 1 中，把“识别”这一词汇标引为处理类概念；在例 2 中，把“关键词”这一词汇标引为词汇类概念；在例 3 中，把“汽车新闻文本”这一短语标引为来源类概念。

3.1.3　概念标引实验

本文对需求文本进行概念标引，本质上是解决一个多分类问题。本文将需求的概念分类任务与训练得到的掩码语言模型相结合，进行小样本学习。通过对需求文本补充任务进行描述，来构建如下的完形填空模板。

长句+待标引词，此为其_____。

上述模板将设计的填空问题作为后缀，在后缀中将概念分类任务转化为完形填空问题。基于构建的概念分类模型，本文将填空的长度统一设置为 2 个字符，完形填空的候选词为{词汇、句子、篇章、处理、领域、来源、方法}。

本文通过在自然语言处理的需求领域语料上对 BERT 进行微调，获得适用于需求领域的 BERT 模型。基于构建的需求领域适配的预训练语言模型，使用 MLM 任务对模型进行训练，采用上述概率计算方式进行实验，以准确率、召回率、F1 值为评价指标。实验结果如表 3 所列。

表 3　概念标引实验结果

Table 3　Conceptual labeling experiment results

标签类别	dev	test
	P/R/F1	P/R/F1
词汇类	1.000/0.978/0.989	0.957/0.936/0.946
句子类	0.920/1.000/0.958	0.880/0.917/0.898
篇章类	0.950/0.905/0.927	0.900/0.837/0.867
处理	1.000/0.953/0.976	0.984/0.953/0.969
领域	0.861/0.940/0.899	0.871/0.794/0.831
来源	0.907/0.886/0.897	0.870/0.909/0.889
方法	1.000/0.953/0.976	0.977/0.977/0.977
平均	0.948/0.945/0.946	0.920/0.904/0.911

本文同时选用其他分类模型在同样的数据集上进行训练，并进行对比实验，以证明本文所提方法的有效性。各模型在测试集上的对比结果如表 4 所列。

表 4　不同模型概念标引结果

Table 4　Results of conceptual citation of different models

模型	*Precision*	*Recall*	*F1*
BERT	0.091	0.111	0.040
ERNIE	0.291	0.179	0.115
Our Method	**0.920**	**0.904**	**0.911**

由结果可知，在使用小样本进行训练时，使用数字对文本标签进行编码，通过 softmax 函数输出标签概率进行预测的方法，无法有效利用标签中包含的语义知识与模型中的领域知识。本文使用构造模板的方法，将概念标签与文本同时进行编码，能够有效利用文本的上下文信息，即模型内部存储的

知识，在标注样本数量较少时，能够有效地进行概念标引。

3.1.4 概念识别结果与分析

为了验证模型对需求文本概念标引结果的有效性，本文选取 3 个需求文本具体实例进行分析，对资源库的资源进行人名识别、发现新闻资源库中人物的言论句，以及统计新闻中的词频的标引结果分别如表 5－表 7 所列。

表 5 模型概念标引结果(例 1)

Table 5 Results of model conceptual citation(example 1)

概念	预测类别	正确类别
资源库的资源	来源	来源
人名	词汇类	词汇类
识别	处理	处理

表 6 模型概念标引结果(例 2)

Table 6 Results of model conceptual citation(example 2)

概念	预测类别	正确类别
新闻资源库	来源	来源
人物言论句	句子类	句子类
发现	处理	处理

表 7 模型概念标引结果(例 3)

Table 7 Results of model conceptual citation(example 3)

概念	预测类别	正确类别
统计	处理	处理
新闻	词汇类	领域
词频	词汇类	词汇类

对比预测类别与正确类别，对于文本中的大部分概念，模型均能正确地进行识别。例 3 中，“新闻”这个概念的正确类别为“领域”，模型错误地将其标引为“词汇类”概念。总体可见，本文提出的需求文本概念标引模型能够有效地对概念的类别进行标引。

3.2 基于需求文本的代码解析模型

本节根据构建的代码库分类模型，确定代码库类别；依据构建的代码检索模型，针对某一细分代码库，将需求文本与代码进行相似度计算，并根据相似度计算结果确定所调用的代码名称，为解析插件进行调用与插件生成奠定基础。具体流程如图 2 所示。

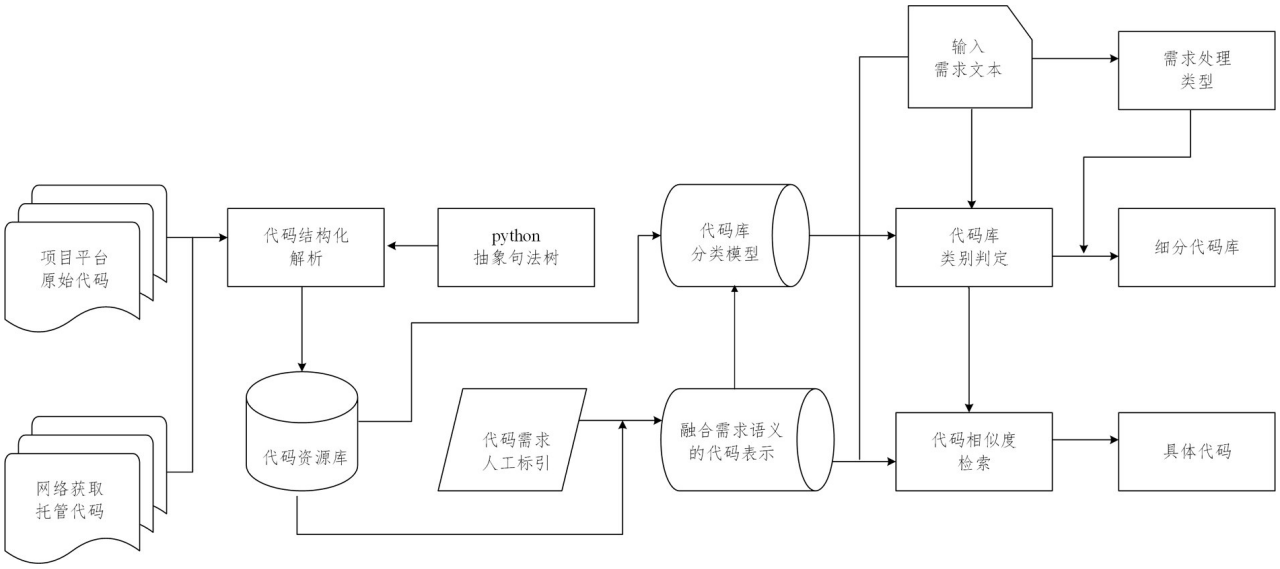

图 2 基于需求文本的代码解析流程图

Fig. 2 Flowchart of code parsing based on equirements text

3.2.1 面向需求文本的代码库分类模型

由于经过人工标引的代码方法文本能够反映一定的语义信息，代码中的单个方法文本中的上下文变量之间存在相互赋值的关系，因此本文使用卷积神经网络的卷积核捕捉代码文本在序列位置上的信息。本文使用 word2vec 词向量模型进行词嵌入，使用卷积核进行特征提取。

本文针对构建的需求概念结构得到的需求处理类型，将代码划分为相应的代码库类型。对于新输入的需求文本，代码库分类模型将其映射到提取类或预处理类代码库，结合前述对需求处理类型的判定，确定最终的细分代码库。具体类型如表 8 所列。

本文按照构建的代码资源库模型，将项目中的算法代码进行结构化解析，构建算法代码资源库。为了丰富算法代码资源，本文所用的原始资源一方面来自于项目实际使用的算法代码，另一方面来自从 Github 代码托管网站获取的算法代码。项目使用的代码如表 9 所列。从代码托管网站 Github 获取的词汇级算法代码如表 10 所列。

本文将需求文本作为输入，调用训练好的代码库分类模型预测需求文本所属的代码库类型。训练结果如表 11 所列。

从模型训练结果来看，模型在准确率方面的表现较好。为了验证模型在需求文本上的泛化能力，本文构建了测试集的需求文本，文本来源于从自然语言处理领域的论文摘要，经过了人工筛选并简要修改。由于获取的论文摘要为中文语料，为了使语料的语义分布与模型的语义分布相近，本文统一对中文语料进行翻译处理，将中文语料翻译成英文，从而获得与模型分布相近的语义信息，并将其作为最终的测试集。测试结果如表 12 所列。通过实验结果可以发现，模型在代码语料的分类上达到了 72%的准确率，在代码库分类任务中取得了一定的效果，验证了模型在面向需求文本的代码库分类任务上具有一定的泛化能力。

经上述实验，可将输入的需求文本映射到代码库类型，结合判定的需求处理类型，确定所属的细分代码库。

表 8　代码库类型

Table 8　Code base types

需求处理类型	细分代码库分类	代码库组成示例
词汇类	词汇预处理类	分词标注、词频统计
	词汇提取类	人名、地名等命名实体识别
句子类	句子预处理类	分句、句子排序
	句子提取类	关键句、事件句、观点句提取
篇章类	篇章预处理类	分段、文本清洗
	篇章提取类	单文档摘要、多文档摘要

表 9　项目现有词汇级算法代码示例

Table 9　Examples of existing vocabulary-level algorithm code for the project

代码库分类	自然语言处理工具名称	工具简介
预处理类	NLPIRSeg	NLPIR 分词工具
	HANLPPostag	HANLP 词性标注工具
	HmmPosTag	基于隐马尔可夫模型的标注工具
	WordFreqUtil	统计词频，并按升序或降序排序
	…	…
提取类	TextRank4zhSubjectExtract	基于 textrank 进行中文主题关键词提取
	HANLPKeywords	使用 HANLP 算法进行关键词提取
	Bilstm_crfNER	调用 bilstm-crf 模型进行人物识别
	LDAKeywordExtract	使用 LDA 算法提取关键词
	…	…

表 10　代码托管网站获取的算法代码示例

Table 10　Examples of algorithmic code obtained from code hosting sites

代码库分类	代码仓库名称	仓库简介
预处理类	wordSegment	Chinese word segmentation algorithm based on entropy
	newword	基于 Nagao 算法统计词频
	single-docterm-freq	count and ranks the frequency keywords
	word_counter	to count words frequency in a text file with nltk
	…	…
提取类	keyphrase_extraction	use tf-idf weighting to rank key phrases
	keyextract	基于机器学习的关键词提取方法
	lgb_predict	利用 jieba 的 tfidf 方法筛选出 Top20 的候选关键词
	Rake_raw	以停用词来切割句子生成候选关键词
	…	…

表 11　代码库分类模型的训练结果

Table 11　Training results of code base classification model

数据集	*loss*	*Precision*
训练集	0.0613	0.9698
开发集	0.8286	0.8744

表 12　代码库分类模型的测试结果

Table 12　Test results of code base classification model

Precision	*Recall*	*F*1
0.728	0.769	0.748

3.2.2　基于分类结果的代码检索模型

本文基于需求语义，将代码库的所有代码进行向量化表示，对代码文档进行检索，实现面向需求文本的算法获取，检索模型构建的具体步骤为：

1）给定业务需求文本，对需求文本进行分词，并进行概念关键词提取；

2）对文本提取结果过滤停用词，获得业务需求的词袋表示；

3）构建代码词袋文档，提取调用的方法名与变量名，将统一代码文档中的变量视为共现关系，保存为共现矩阵 2；

4）根据共现矩阵 1 和矩阵 2，计算需求向量与代码文档向量的相似度，检索分值的计算式如式(3)所示：

$$score(q,j)=\frac{\boldsymbol{x}_j \cdot \boldsymbol{x}_q}{\max(\| \boldsymbol{x}_q \|_2 \cdot \| \boldsymbol{x}_j \|_2,\varepsilon)} \tag{3}$$

其中，$\boldsymbol{x}_q$ 为需求文本即查询 query 的向量表示，$\boldsymbol{x}_j$ 为算法代码 j 的向量表示。

在面向需求的代码检索中，检索文本为人工输入的自然语言，其输入为面向业务的需求实现描述，通过对需求描述的向量化表示，从算法代码库中检索所有该类型的代码，并选择排序最高的方法名作为最终插件生成的调用代码。本节中，本文以上述需求为研究对象，将给定的业务需求输入检索模型并与人工构建的排序结果进行比较，从而评定模型获取所需代码的能力。

本文基于项目实际需求，构建了需求文本，并使用构建的检索模型对同一个需求类型下的所有代码进行检索。需求示例如下：

query：“对文本进行人名识别”。

在实际检索中，本文构建的代码库的代码语言为基于英文的 python 代码，因此本文首先将 query 转为英文后再进行检索。

对于模型的检索结果，本文使用前 K 准确率(percision@K)作为评价指标。最终检索结果分值如表 13 所列。

$$percision@K=\frac{\text{前 } K \text{ 个结果中准确的个数}}{K} \tag{4}$$

表 13　需求文本检索结果

Table 13　Requirements text retrieval results

需求文本	query	score
检索排序	bilstm_crfNER	0.83242700
	pyhanlpNER	0.82986070
	loc_ch	0.74548864
	loc	0.74361860
	ltpNER	0.71313150
percision@5	60%	

分析结果可知，对于“对文本进行人名识别”这一需求，返回的结果中排在首位的能够满足需求，但 loc_ch.py 和 loc.py 两个用于地名提取的代码排在人名提取代码 ltpNER.py 前面。前 5 个检索结果的准确率达到 60%。需求文本检索实验结果证明，本文构建的需求检索模型能够有效面向需求文本，对该需求处理类型下的算法代码进行检索。

4　资源按需解析关键技术与评价

本文通过从业务需求文本到代码库的构建，最终生成插件代码的资源按需解析模型。针对输入的需求文本进行

解析，判定需求处理类型及所属的代码库；针对所属的代码子库，根据构建的检索模型检索到目标代码，实现资源的按需解析。本章基于上述流程，通过需求案例实现资源的按需解析，验证资源按需解析模型的有效性。

4.1 需求文本概念解析结果分析

本文向项目组业务人员收集一定数量的需求文本作为实验数据，共构建了 6 条需求文本，如表 14 所列。

表 14 人工收集的需求文本

Table 14 Manually collected requirements text

编号	需求文本
query1	对政策文本进行人名和地名识别
query2	对正文进行分词标注、实体抽取、词频统计关键词提取等分析
query3	使用 BiLSTM-CRF 模型识别中医临床症状术语
query4	针对网络舆情的评论、回复进行分词
query5	从文本评论中提取用户观点
query6	从新闻文本中提取主题句

短语抽取与分词结果如表 15 所列。

表 15 需求文本短语抽取与分词结果

Table 15 Results of demand text phrase extraction and segmentation

编号	分词结果
query1	对/政策文本/进行/人名/和/地名/识别
query2	对/正文/进行/分词/标注/、/实体/抽取/、/词频/统计/等/分析
query3	使用/BiLSTM-CRF 模型/识别/中医/临床/症状术语
query4	针对/网络舆情的评论/、/回复/进行/分词
query5	从/文本评论/中/提取/用户观点
query6	从/新闻文本/中/提取/主题句

本文对分词后的文本对词汇逐个进行概念标引，标引结果如表 16 所列。

表 16 需求文本概念标引结果

Table 16 Results of requirements text concept citation

编号	标引	概念	标引	概念	标引	概念
query1	来源处理	政策文本识别	句子	人名	词汇	地名
query2	篇章 词汇处理	正文 实体统计	词汇处理	分词抽取 分析	处理 词汇	标注 词频
query3	方法 领域	BiLSTM-CRF 模型 临床	处理 词汇	识别 症状术语	领域	中医
query4	领域 方法	网络舆情 分词	句子	评论	方法	回复
query5	句子	文本评论	处理	提取	句子	用户观点
query6	领域	新闻文本	处理	提取	句子	主题句

概念标引的错误主要集中在“需求”类概念下的“词汇类”“句子类”和“篇章类”概念。总体来看，概念能够将训练语料中的绝大部分概念进行正确标引，有效地对需求文本进行解析。

4.2 需求文本概念结构解析结果分析

针对概念标引结果，根据概念结构先验规则，解析需求文本的概念结构，随即获得相应概念结构所属的需求处理类型。需求处理类型判定结果如表 17 所列。

受限于概念先验关系提取算法，在需求文本 4 中，未得到“提取”与“观点”这两个处理类概念与句子类概念之间的概念关系。综合所有结果来看，本文模型能够较为有效地提取出需求文本的概念结构。

表 17 需求文本概念结构解析结果

Table 17 Parsing results of requirements text conceptual structure

编号	关系	概念 1	概念 2	预测需求处理类型	实际需求处理类型
query1	词汇类处理 处理对象	地名 政策文本	识别 识别	词汇类处理类型	词汇类处理类型
query2	词汇类处理 词汇类处理 词汇类处理 词汇类处理	分词 实体 词频 词频	标注 抽取 统计 分析	词汇类处理类型	词汇类处理类型
query3	处理方法 词汇类处理 处理领域 处理领域	BiLSTM-CRF 模型 症状术语 中医 临床	识别 识别 症状术语 症状术语	词汇类处理类型	词汇类处理类型
query4	未提取出概念对			判定失败	词汇类处理类型
query5	句子类处理	文本评论	提取	句子类处理类型	句子类处理类型
query6	句子类处理 处理对象	提取 新闻文本	主题句 提取	句子类处理类型	句子类处理类型

4.3 面向需求的代码库分类结果分析

对需求文本解析完毕后，本文面向需求文本进行代码库分类。代码库分类结果如表 18 所列。

表 18 面向需求的代码库分类结果

Table 18 Requirements-oriented code base classification results

编号	代码库分类结果	Score	判定细分代码库	实际细分代码库
query1	提取类	0.68497510	词汇提取类	词汇提取类
query2	提取类	0.68497510	词汇提取类	词汇提取类
query3	提取类	0.84659153	词汇提取类	词汇提取类
query4	—	—	判定失败	词汇预处理类
query5	提取类	0.79229990	句子提取类	句子提取类
query6	提取类	0.84659153	句子提取类	句子提取类

由于需求文本 4 在概念结构解析中未解析出相应的需求处理类型，因此导致资源按需解析流程中止，未能进一步对细分代码库分类。从整体细分代码库判定结果来看，本文模型能够一定程度上对细分代码库进行正确判定。

4.4 基于分类结果的代码检索结果分析

获得指定的细分代码库后，针对相应库下的代码进行检索，检索结果如表 19 所列。

表 19 基于分类结果的代码检索结果

Table 19 Code retrieval results based on classification results

编号	代码检索结果	Score	最终是否满足业务需求
query1	Loc_ch	0.76444240	部分满足，未满足地名识别需求
query2	bilstm_crfNER	0.84152320	部分满足，未满足分词标注、词频统计需求
query3	ltpNER	0.76055450	是，未使用 bilstm_crf 方法
query4	—	—	否
query5	parse_news	0.78580606	是
query6	parse_news	0.73091570	是

需求文本 4 的错误结果已在前文说明，故不再赘述。针对需求文本 1 与文本 2，实际上在一条文本中包含了多个需求，本文的研究自始至终为针对单条需求文本中的单个需求进行解析，因此能够分别满足需求文本 1 与文本 2 中的其中一个需求。从最终目标代码的检索结果来看，本文模型能够

一定程度上将需求文本关联到待组配的目标代码。

对比单纯使用清华大模型 ChatGLM 对需求文本进行代码生成，本文使用的方法大多聚焦于具体业务领域内，立足于团队开发的相关业务平台，而大模型生成的代码只能完成比较基础且宽泛功能的实现，大多不能满足或者只能部分满足业务需求，需要后续的完善工作，无法针对业务需求进行快速的处理。

4.5　基于检索结果的解析插件生成与激活

本节基于资源按需解析流程进一步构建插件代码结构模型，用于关联基于需求匹配的 NLP 代码与插件代码，并提出了基于上述模型的外部代码插入方法，实现插件代码的生成。解析插件更新以前台输入需求文本为触发，连通需求分类、代码库分类、检索等核心组件，以成功启动资源解析插件为终点，构成资源按需解析系统完整闭环。本文以将地名识别算法插入新闻资源解析插件为例，介绍基于动态编译代码更新的资源按需解析。

资源解析插件遵循统一的开发规范，将代码名称插入插件指定位置，这需要对资源解析插件的结构进行分析，厘清插件内部结构与业务逻辑，为插件代码的生成奠定基础。目前团队共开发了 93 个资源解析插件，运行在项目后台中。主要资源解析插件示例如表 20 所列。

表 20　主要资源解析插件示例

Table 20　Examples of main resource parsing plugin

资源解析插件名称	资源名称	插件描述
event	事件	简报生成事件
docbook	书籍	读取 word 格式的文本
LocalPolicyBank	地方政策库	解析爬取的地方政策网站资源
MetaEvent	新闻元事件库	从新闻元事件资源解析
newReport	周报	解析爬取的资源生成周报
newsBaigong	新闻	excel 格式网页解析
technicalReference	科技参考	对科技参考实体类进行加工处理形成科技参考再生资源
regenerateReport	再生报告	新型简报生成，依据模板生成简报
review	综述	依据主题词、文献集合、结构模板生成综述
workReport	工作报告	txt 格式的工作报告解析
…	…	…

资源解析插件代码生成需要对业务流程与基本结构进行拆解分析，本文将其分为 3 个部分，即字段定义、字段封装与处理、资源生成。字段定义中的字段类文件封装了基于 schema 生成的实体类；字段封装与处理中的字段封装工具，数据读取与处理工具实现了从数据库读取并赋值到实体类；资源生成部分实例化了以上各个类，并调用 SchemaUtil 类中的 buildXml 方法生成结构化的 xml 文件，完成资源的结构化解析。解析插件业务流程如图 3 所示。

在前台输入需求文本后，基于需求的代码匹配结果，将算法名称插入插件指定位置。写入完毕后，将插件代码进行打包运行，查询配置文件中的算法调用地址进行解析。

经过上述资源按需解析流程，获得了待组配的目标算法代码名，随后按照解析插件的外部代码调用流程进行配置。待组配的算法代码名在 Java 代码中以变量的形式存在，需要使用指定方法加载，经过编译后调用。本文使用 Java 存在的反射机制，在程序运行时实现插件更新。

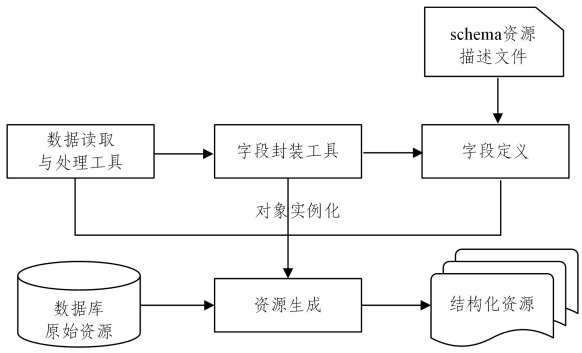

图 3　解析插件业务流程

Fig. 3　Business process of parsing plugin

解析插件实现更新后启动解析，将资源按照需求实现对应字段的提取与入库。以 3 月 1 日爬取的新闻为例，在新闻资源库中查询到 1 005 条结果，地名提取工具提取的地名已保存在 news_info_country 字段中，如图 4 所示。

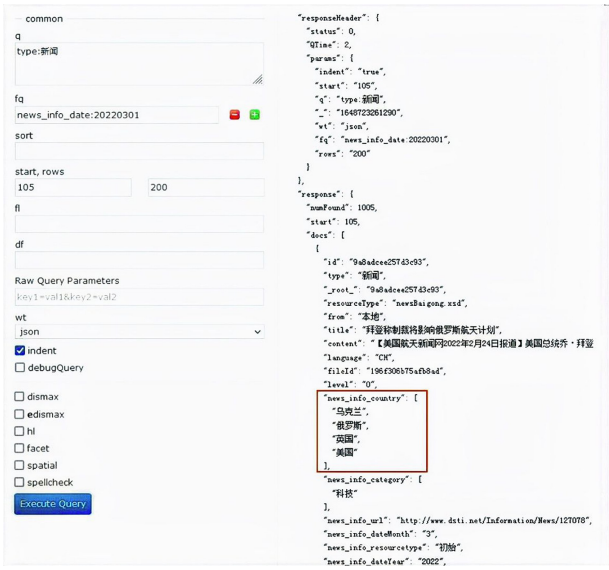

图 4　新闻资源地名提取解析结果

Fig. 4　Results of extraction and parsing of news resources geographical names

结束语　本文以业务需求与资源解析为研究对象，结合自然语言处理技术，在需求文本与代码资源进行结构化解析的基础上，通过需求与代码建模实现了资源的按需解析，针对业务需求实现解析插件的控制式生成，提高软件的复用与开发效率。在今后的研究中，本项目会尝试利用外部知识辅助需求文本进行概念标引，扩展与补充代码库的类别与数量，充分利用所使用到的代码仓库描述文档、代码注释反映的代码功能语义等。

参 考 文 献

[1]　LAMPLE G, BALLESTEROS M, SUBRAMANIAN S, et al. Neural architectures for named entity recognition[J]. arXiv: 1603.01360, 2016.

[2]　DEVLIN J, CHANG M W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv: 1810.04805, 2018.

[3]　LECUN Y, BOTTOU L. Gradient-based learning applied to do-

cument recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
[4] BENGIO Y, SIMARD P, FRASCONI P. Learning long-term dependencies with gradient descent is difficult[J]. IEEE transactions on neural networks, 1994, 5: 157-166.
[5] KIM Y. Convolutional neural networks for sentence classification[J]. arXiv: 1408.5882, 2014.
[6] SHI B, XIANG B, CONG Y. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2016, 39(11): 2298-2304.
[7] HOCHREITER S, SCHMIDHUBER J. Long Short-Term Memory[J]. Neural Computation, 1997, 9: 1735-1780.
[8] GERS F A, SCHMIDHUBER J, CUMMINS F A. Learning to Forget: Continual Prediction with LSTM[J]. Neural Computation, 2000, 12: 2451-2471.
[9] GRAVES A, SCHMIDHUBER J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J]. Neural Networks, 2005, 18(7): 602-610.
[10] CHUNG J, GULCEHRE C, CHO K, et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling[J]. arXiv: 1412.3555, 2014.
[11] PETERS M, NEUMANN M, IYYER M, et al. Deep Contextualized Word Representations[C]// Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1(Long Papers). 2018: 2227-2237.
[12] RADFORD A, NARASIMHAN K, SALIMANS T, et al. Improving language understanding by generative pre-training[EB/OL]. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
[13] LU J, YING W, SUN X, et al. Interactive Query Reformulation for Source-code Search with Word Relations[J]. IEEE Access, 2018, 6: 75660-75668.
[14] MCMILLAN C, GRECHANIK M, POSHYVANYK D, et al. Portfolio: finding relevant functions and their usage[C]// Proceedings of the 33rd International Conference on Software Engineering(ICSE 2011). Waikiki, Honolulu, HI, USA, 2011(5): 111-120.
[15] LV F, ZHANG H, LOU J, et al. Codehow: Effective code search based on API understanding and extendedboolean model[C]// 30th IEEE/ACM International Conference on Automated Software Engineering. ASE 2015, Lincoln, NE, USA, 2015: 260-270.
[16] RAHMAN M M, CHANCHAL R. Nlp2api: Query reformulation for code search using crowdsourced knowledge and extra-large data analytics[C]// 2018 IEEE International Conference on Software Maintenance and Evolution(ICSME). IEEE, 2018: 714-714.
[17] HUSAIN H, WU H, GAZIT T, et al. Codesearchnet challenge: Evaluating the state of semantic code search[J]. arXiv: 1909.09436, 2020.
[18] GU X, ZHANG H, KIM S. Deep code search[C]// Proceedings of the 40th International Conference on Software Engineering. ICSE 2018, 2018: 933-944.
[19] YIN P, NEUBIG G. A syntactic neural model for general purpose code generation[C]// Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics(ACL 2017) Vancouver, Canada, Volume 1: Long Papers, Association for Computational Linguistics. 2017: 440-450.
[20] ZHANG J, WANG X, ZHANG H, et al. A novel neural source code representation based on abstract syntax tree[C]// Proceedings of the 41st International Conference on Software Engineering(ICSE 2019). Montreal, QC, Canada. IEEE / ACM, 2019: 783-794.
[21] WAN Y, SHU J, SUI Y, et al. Multi-modal attention network learning for semantic source code retrieval[C]// 34th IEEE/ACM International Conference on Automated Software Engineering(ASE 2019). San Diego, CA, USA. IEEE, 2019: 13-25.
[22] YANG H. BERT meets chinese word segmentation[J]. arXiv: 1909.09292, 2019.
[23] SCHICK T, SCHÜTZE H. Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference[C]// Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. 2020: 255-269.

LIU Yao, born in 1972, Ph. D, researcher, is a distinguished member of CCF (No. 17606D). His main research interests include natural language processing, knowledge organization, and knowledge engineering

(责任编辑: 何杨)