华北计算技术研究所

研究生毕业论文开题报告

**基于深度学习的程序生成模型**

**研究与设计**

**Research and Design of Program Generation Model Based on Deep Learning**

学生姓名： 赵东杰

指导教师： 仇建伟

部 门： 系统一部

专 业： 计算机应用技术

2019年10月

目 录

[1 选题背景和意义 1](#_Toc497463844)

[1.1 选题背景 1](#_Toc497463845)

[1.2 选题意义 1](#_Toc497463846)

[2 国内外发展动态 3](#_Toc497463847)

[2.1 智能CAD相关研究 3](#_Toc497463848)

[3 研究目标 6](#_Toc497463850)

[4 研究内容 6](#_Toc497463851)

[5 关键技术 9](#_Toc497463855)

[6 论文实施计划 11](#_Toc497463859)

[7 参考文献 12](#_Toc497463860)

# 选题背景和意义

## 选题背景

随着深度学习在2006年正式提出，标志着人工智能进入了一个新的阶段，其中深度神经网络在各个领域的实际应用，也极大的促进了人们生活的智能化。特别是数据量和计算机计算能力快速发展的背景下，神经网络在诸多学术领域变成最先进的技术，并且已经成功地在生产中得到部署，在自然语言处理、图像处理和语音处理等方面都得到了广泛的应用。然而作为诸多应用背后的开发人员，编写代码是基本功，但是编写冗长的代码也极大的消耗了开发者的耐心，深度学习却没有为程序员的基础开发工作带来实质上的改进。

部门项目组为了满足部门综合业务的开发需求，设计并实现了综合业务可视化开发平台，满足了开发人员快速构建系统架构的需求。但是对于最基础的编码过程，综合开发平台相对于通用的开发环境并没有很大提升。因此，提高开发过程中代码编写的自动化、智能化程度就成为项目组目前需要解决的问题，其中最主要的问题在于提高代码补全中的准确性和完整性，以及根据自然语言生成相应代码两方面。

## 选题意义

现代集成开发环境(Integrated Development Environment,简称IDE)为程序员提供了基本的拼写错误检查，预测函数名、关键字、方法等基本功能，通常给出的预测信息都是以提示的方式给出，并且按照字母排序的方式排列，这往往不符合程序员本身的意图，更多的时候，开发人员希望给出的预测是根据与代码功能的匹配度排序，从而能够更快的进行选择。

同时，开发过程中的文档注释，往往包含了开发人员在设计中的诸多信息，如果能够利用开发人员给出的自然语言的注释信息，自动生成能够实现开发人员意图的代码，对于提高开发人员的开发效率，以及促进整个业务的开发进程，都会有很大的作用。

因此，本文希望基于深度学习模型，通过对开发人员已有的代码以及不断开发的代码进行学习，为开发人员提供更符合自身开发习惯、更加规范、以及更加智能化的的程序生成服务，从而提高开发人员的开发效率以及开发规范性，加快项目开发进度，促进部门整体业务的快速发展。

# 国内外发展动态

代码补全是现代IDE的重要组成部分，也是开发人员判断一个IDE是否好用的重要标准，同时代码补全也是程序生成最常见的技术。通过帮助开发人员预测方法名、关键字、属性等，代码补全对下一个token给出预测，并以字母序进行排序，增加了开发人员选择的时间。传统代码补全方法主要基于两方面：一是根据利用静态类型信息，加上设计的各种启发式的规则来决定预测的token，如Eclipse通过静态的类别信息给开发人员推荐方法，其中候补信息通过字母进行排序；二则是利用代码样例以及语义信息来进行token的补全，如2010年Huo等人通过称为BBC的技术，该技术通过对API进行排序和筛选，从而提高Eclipse本身基于类型的代码补全效果。但这两种传统的方法一般都要求人为的设计启发式的规则，因此限制了方法本身的发展。

深度学习的出现，改变了传统的代码补全的方式，通过从大量的代码中进行学习，深度学习可以了解到代码token之间的概率分布，从而基于学习到知识，提高token预测的准确率。目前通过深度学习进行代码补全的主要流程如下：

1. 训练阶段：通过从开源社区或者Github等开源代码库获取大量语料库，并通过代码解析器对源代码进行处理，如将代码转化为语法树或者token序列，之后选择一个合适的深度神经网络模型，如语言模型，对语料库中的数据进行训练。而目前效果最好的语言模型包括BERT、GPT以及GPT-2等。
2. 代码补全阶段：在当前需要补全的的位置调用训练好的模型，该模型根据当前已经输入的代码片段来预测需要补全的token。

其中主要工作包括：基于程序序列化特征的补全，如Hindle在2012提出的利用语言模型学习程序的顺序概率特征，并据此对下一个token进行预测；基于程序局部性特征的补全，如Tu等人提出通过在已有的语言模型的基础上，添加缓存机制，从而来维护程序代码的局部信息；基于程序结构化特征的补全：如Liu等人通过遍历语法树，对得到的语法树序列进行建模，或者Raychev等人直接利用树结构对程序的语法树进行建模。

基于功能描述的程序生成，体现了从自然语言到程序语言的翻译过程。而自然语言的文本化以及文本对应的二义性，程序代码的复杂结构以及多样性，使得自然语言到程序语言的翻译过程成为一个难题。而深度学习正好可以用来解决这个问题。

Betagy等人利用深度学习从自然语言生成IFTTT代码。IFTTT是If-This-Then-That的简写，相对于普通程序来说，IFTTT程序结构更加简单也更加容易学习该程序的结构规则。Gu等人利用深度学习提出了DEEPAPI，一个根据自然语言生成API序列的工具，从而能够生成通用程序设计语言比如Java、Python等代码片段。Cai等人通过将传统查询解析方法与深度学习方法相结合的方式提高了SQL生成的准确率。

目前程序生成的深度学习模型主要包括两种，一种是基于语言模型，另一种是基于机构化代码生成网络。语言模型通常直接将解析过的代码当做文本作为输入，对文本的概率分布来进行建模。常见的语言模型有N-gram模型和循环神经网络(Recurrent Neural Network,简称RNN)。其中N-gram可以有效的学习代码的上下文，但是对代码的语义信息却无能为力。而RNN可以捕捉句子中词与词之间的规律，因此许多N-gram与RNN结合的模型，以及RNN的变体都相继被提出并得到了广泛的应用，包括Raychev等人通过将N-gram与RNN结合的方法，在Java中进行API级别的补全，以及长短记忆模型(Long Short-Term Memory,简称LSTM)等。结构化代码生成网络则是通过对代码的强结构性进行结构化的建模，具体又分为基于语法树进行建模以及基于图网络进行建模。如Dong等人通过提出一种树解码器SEQ2TREE，将编码器的结果通过RNN生成相应的token。而Allamanis等人通过图的网络来表示代码结构和语义特征。

高质量的代码数据集是进行深度学习的前提，在图像和自然语言处理处理方面都有各自公开的数据集，如图像处理的ImageNet，而在程序语言中，这方面的数据集非常少。目前已有的一些公开的数据集包括：Xing等人提供的从GitHub中抽取的Java方法以及相应的JavaDoc描述，Bhoopchand等人提供的GitHub上fork数前949以及星级大于100的Python项目。目前这些数据集大多来自于GitHub，同时为了保证代码质量，都会对star数以及fork数目进行一定的限制，从而抽取较高的代码片段加入数据集之中。

目前国内外已经实现商用的基于深度学习的只能代码补全工具包括Kite、TabNine，以及aiXcoder。其中Kite是由来自MIT和斯坦福的开发人员组成的团队，主要是针对Python语言的智能开发插件，它提供了整行代码补全，代码片段补全以及直接在编辑器中提供开发文档等功能，来加速Python代码的开发速度。而TabNine主要是由来自加拿大的大四学生Jackson开发完成，并在主流开发环境中都已上线，作者是基于OpenAI提供的GPT-2模型，并通过GitHub上上百万的代码文件对模型进行优化，从而实现了对多达23种编程语言的支持。而aiXcoder的开发人员来自于北京大学高可信软件技术教育部重点实验室，aiXcoder融合了基于序列的程序代码语言模型、基于抽象语法树和程序逻辑关系的图神经网络等方法，共同打造了一个完整的系统，从而能够更准确地进行代码生成。同时aiXcoder也主动与其他两款应用进行了对比，结果显示aiXcoder可以用更少的键盘输入次数来实现相同的功能。而在使用方面，因为Kite只针对Python语言，所以只有单一的版本。aiXcoder和TabNine都是基于多种语言的，其中aiXcoder根据不同的语言实现了不同的模型，从而提供不同的版本，而TabNine因此是基于通用语言模型进行开发，所以对于不同的语言，都提供了统一的版本。

# 研究目标

基于所内创新基金项目——综合业务信息系统可视化开发平台，通过预训练通用语言模型与基于抽象语法树的结构化网络相结合的方法，研究并设计程序生成模型，为部门开发人员提供更加准确、智能化的程序生成服务，提供开发效率。

# 研究内容

## 基于数据关联的意图捕捉和辅助设计技术研究

## 基于领域知识的意图捕捉和辅助设计技术研究

1. 基于领域知识的设计意图捕捉技术

## 体系结构智能辅助设计原型系统的设计与实现

分析体系结构辅助设计的多种意图捕捉方法，掌握相关意图捕捉方法和捕捉

# 关键技术

## 基于数据关联的意图捕捉技术

在进行体系结构辅助。

## 基于领域知识的意图捕捉技术

在进行知识提示，这也是本课题需要解决的关键技术之一。

## 基于Agent的智能辅助设计实现技术

进行体系键技术之一。

# 论文实施计划

毕业设计任务时间划分为五个阶段：

|  |  |
| --- | --- |
| 第一阶段  （2019.09.01-  2019.10.31） | 收集整理资料，对毕设题目进行确定，对整个课题研究进行初步的了解和掌握。对收集的资料进行分类、阅读。归纳相关文献。 |
| 第二阶段  （2019.11.01-  2020.05.31） | 通过收集整理材料，对相关数据进行处理，对所收集的材料进行系统性的学习。充分认识相关研究课题的关键难点问题。在学习的过程，完成小论文的发表。 |
| 第三阶段  （2020.06.01-  2020.10.30） | 对相关模型进行实现。 |
| 第四阶段  （2020.11.01-  2021.01.30） | 对模型进行测试与验证。 |
| 第五阶段  （2021.02.01-  20121.03.31） | 完成毕业设计和毕业论文。 |

# 参考文献

[1] Bass L, Clements P, Kazman R. Software architecture in practice[M].Boston:Addison Wesley,2003