NEURAL CODE COMPLETION 未完

ABSTRACT

Code completion, an essential part of modern software development, yet can be challenging for dynamically typed programming languages. In this paper we explore the use of neural network techniques to automatically learn code completion from a large corpus of dynamically typed JavaScript code. We show different neural networks that leverage not only token level information but also structural information, and evaluate their performance on different prediction tasks. We demonstrate that our models can outperform the state-of-the-art approach, which is based on decision tree techniques, on both next non-terminal and next terminal prediction tasks by 3.8 points and 0.5 points respectively. We believe that neural network techniques can play a transformative role in helping software developers manage the growing complexity of software systems, and we see this work as a first step in that direction.

1 INTRODUCTION

As the scale and complexity of modern software libraries and tools continue to grow, code completion has become an essential feature in modern integrated development environments (IDEs). By suggesting the right libraries, APIs, and even variables in real-time, intelligent code completion engines can substantially accelerate software development. Furthermore, as many projects move to dynamically typed and interpreted languages, effective code completion can help to reduce costly errors by eliminating typos and identifying the right arguments from context.

However, existing approaches to intelligent code completion either rely on strong typing (e.g., Visual Studio for C++), which limits their applicability to widely used dynamically typed languages (e.g., JavaScript and Python), or are based on simple heuristics and term frequency statistics which are often brittle and are relatively error-prone. In particular, Raychev et al. (2016a) proposes the state-of-the-art probabilistic model for code, which generalizes both simple n -gram models and probabilistic grammar approaches. This approach, however, examines only a limited number of elements in the source code when completing the code. Therefore, the effectiveness of this approach may not scale well to large programs.

有效性不适用于大型程序

In this paper we explore the use of deep learning techniques to address the challenges of code completion for the widely used and dynamically typed JavaScript programming language. We formulate the code completion problem as a sequential prediction task over the traversal of a parse-tree structure consisting of both non-terminal structural nodes and terminal nodes encoding program text. We then present simple, yet expressive, LSTM-based (Hochreiter & Schmidhuber (1997)) models that leverage additional side information obtained by parsing the program structure.

Compared to widely used heuristic techniques, deep learning for code completion offers the opportunity to learn rich contextual models that can capture language and even library specific code patterns without requiring complex rules or expert intervention.

We evaluate our recurrent neural network architecture on an established benchmark dataset for the

JavaScript code completion. Our evaluations reveal several findings: (1) when evaluated on short

programs, our RNN-based models can achieve better performance on the next node prediction tasks

compared to the prior art (Bielik et al. (2016); Raychev et al. (2016a)), which are based on decisiontree

models; (2) our models’ prediction accuracies on longer programs, which is provided in the test

set, but were not evaluated upon by previous work, are better than our models’ accuracies on shorter