卒業論文あるいは修士論文等の報告書
Overview of bachelor's thesis or master's thesis

氏名/Name YUE, Ziran 日付 /Date 2024年09月10日

論文指導担当者 Supervisor in charge	教員氏名/Name XU, Yanyan 職位/Title Associate Professor 所属大学/University Beijing Forestry University E-mail or phone xuyanyan@bjfu.edu.cn
論文題目 Topic of thesis	Research and Implementation of Python Code Auto-completion Engine Based on Deep Learning

卒業論文あるいは修士論文について、(1)背景と課題、(2)目的と目標と新規性、(3)あなたが実施した方法と工夫、(4)得られた結果およびその評価を記述してください。**卒業論文が課せられていない場合は**、次のどちらかを記述してください。プロジェクト研究等における上記(1)～(4)、または関心を持って学習した(1b)技術・科目、(2b)関心を持った理由、(3b)どのように取り組んだか、(4b)得られた知識や能力を記述してください。あなたにしか書けないことを中心に論理的かつ具体的に記述し、能力や成果をアピールする部分には下線をしてください。論文、口頭発表等があれば、その著者・題名・掲載誌・発表年月日等も記入してください。(本用紙枠内 1000 字以内厳守。但し、論文・口頭発表等の記述は文字数としてカウントしない。別紙記入不可。印字可)

記述内容は ☑卒業論文、修士論文 ☐プロジェクト等 ☐学習 (一つ選んでチェックしてください。)

Summarize your bachelor's thesis or master's thesis especially (1) background and challenge, (2) objectives, goals, and originality, (3) proposed method and your contributed idea, and (4) results and evaluation. **If you are not required to write a graduation thesis**, you have two choices. Either describe a project research you have taken by detailing (1) to (4) as mentioned above, or describe a technology or subject you have learned which drew your interest: (1b) technology or subject, (2b) reason why you are interested, (3b) how you have handled the subject, and (4b) what knowledge or skill you acquired.

Write from your own perspective logically and concretely. Underline the parts which exhibit your personal competence and the results obtained thanks to you. If you published journal papers or made oral presentations, write the author(s), title, journal in which it was published and date of publication. (Please limit your essay to 500 words on this form. References are not included in the count. Do not attach any separate sheet. Printing is acceptable.)

☑Bachelor's thesis, master's thesis ☐Project research ☐Learning, is described (Select one and check it.)

1. Background and Challenge

In software development, code completion (Figure 1) is an indispensable feature in modern code editors. By suggesting potential code based on the current context, code completion speeds up the programming process. Traditional code completion engines analyze the content and structure of code in a static way. However, it is still challenging to provide semantically correct code while maintaining the corresponding code style, given that static analysis is restricted to words explicitly present in the context.

2. Objectives, Goals and Originality

Our goal is to propose a model for Python that predicts context-aware, semantically relevant code. We will propose an evaluation method to measure the performance of our approach and an LSP (Language Server Protocol)-based completion engine will be developed for testing in a production environment. Our model aims to outperform traditional methods, particularly in out-of-vocabulary (OOV) scenarios.

3. Methods and Contributions

Data Preparation: Code snippets are crawled from the internet and then formatted into a consistent style. Comments, indents, and dedents are removed to ensure a clean, unified input for our model.

Model: We developed a character-level model (Figure 2) for this task. After tokenizing the input, an embedding layer encodes the input into a character-level space. This encoded input is then passed through a long short-term memory neural network (LSTM), allowing the network to understand the input flow in a sequential manner. Finally, a fully connected (FC) layer transforms the output back into human-readable text.

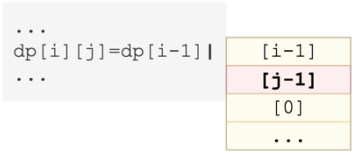

Figure 1. Code completion task

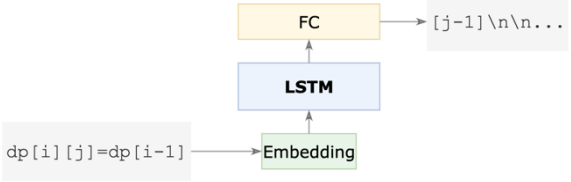

Figure 2. Character-level model

Evaluation Method: We proposed an evaluation method (Eq 1) using the Levenshtein distance (edit distance, which measures the minimum number of steps required to convert one string into another by adding, removing, or changing characters). The accuracy of our model on a given file (Eq 2) is calculated as the "average accuracy" across all possible completion positions within that file where K denotes the number of lines while N stands for maximum possible length at given position.

$$\mathcal{L} = 1 - \frac{Levenshtein(Str_{pred}, Str_{src})}{\max(Len_{pred}, Len_{src})}$$

Equation 1. Accuracy on certain position

$$Text_{acc} = \frac{\sum_{k=0}^{k<K} \sum_{n=1}^{n\leq N} \mathcal{L}_{k,n} \times \frac{log(n)}{log(n!)}}{K}$$

Equation 2. Accuracy across one sample

4. Results and Evaluation

We tested our model on various LSTM variants, finding that increased complexity led to higher accuracy, suggesting room for further improvement. The model showed instability, possibly due to random dropout layers. Despite these limitations, our model was able to generate outputs for OOV cases.

We also implemented a code completion engine based on our model and integrated it with the LSP in Visual Studio Code. While this implementation does not directly impact model performance, it provides an accessible way to interact with our work and similar approaches through a more visual, user-friendly interface.