IntelliCode Compose: Code Generation using Transformer

MohammadAmin Raeisi

Summary
This paper proposes a general-purpose code completion framework, named IntelliCode Compose, capable of generating code sequences of arbitrary token types, including local variables, methods or APIs, arguments, punctuation, language keywords, and delimiters. The proposed system can learn to infer types of programming language identifiers and long-range code semantics without inputs extracted using a static analyzer explicitly passed to the model as features. They used transformer models with attention mechanisms to train data.

It uses a statistical language modelling approach based on the n-gram model as a baseline in this work. The n-gram model is the probabilistic Markov chain model for predicting text given the context of n-1 preceding tokens.

A naive beam search implementation would iterate over the top k candidates at every step to produce the output vector. Given sequential nature of the beam search decoding, it cache the attention keys and values of the transformer blocks as computed by the model for previous step (token), passing it as input to the model at the current step instead of recalculating from scratch.

Strengths

- The tool they have presented is very useful in the daily life of programmers.
- Its use is not limited to a specific programming language and supports many languages.
- They have used the latest methods of artificial intelligence to generate code.

Weaknesses

- Understanding the paper's content is difficult for those unfamiliar with neural networks and transformers.
- The introduction section is concise and could have explained the problem and its solution more by providing an example.
- If the paper use the combination of formal methods with machine learning, the answer would probably reach faster.
- The paper did not explain the details of the model structure and the presented algorithm but explained the generalities.