Borealis AI - Let's Solve It Undergraduate Mentorship Program

Java-to-Python Transcompiler Extension

Olivia Xu, Nadia Enhaili, Anoushka Singhal, and Maisha Thasin

December 1, 2021

1 Introduction

As students and aspiring programmers, we found that we often had to juggle various languages for short time periods. When faced with using a language we had not used in a while, we were inevitably slower and less efficient than with languages we regularly used. This loss in efficiency could mainly be attributed to confusion between the language we were using and the one we were habituated to - for example, trying to call a funtion in language A the way it is called in language B.

From the start, we were leaning towards some form of translator tool but we hadn't decided on the specifics yet. In the brainstorming phase, we came across CodeGen, Facebook AI's transcompiler. A transcompiler, also known as source-to-source translator, is a system that converts source code from a high-level programming language (such as C++ or Python) to another. [1]. CodeGen supports C++, Java and Python and outputs highly accurate compilable code.

Given the scope of the mentorship program, time constraints, and our individual skillsets, we decided to build a similar albeit much simpler tool that would solve our immediate problem.

Keywords
transcompiler, translator, Facebook Research, programming

2 Materials & Methods

The first step was trying out the CodeGen models to see what we were working with. We immediately ran into trouble with that, as missing dependencies caused several errors. With great help from our mentor, we ended up with a working Google Colaboratory environment and were able to test out a few code chunks and keywords. We quickly noticed that the translations were not consistently accurate - sometimes outputting perfectly compilable code, and other times nonsense or even nothing at all.

Around this time, we decided that we wanted our end product to be a Visual Studio Code extension. Our vision was that it would suggest the appropriate correction when it detected wrong syntax or keywords. We first built a mock VSCode extension to get a feel for how it works and understand its limitations. From there, we concluded that we needed a list of "bad" keywords and their corresponding "good" correction - for example, the keyword *def* would be considered "bad" if you were coding in Java, and the correction would be *public static void main.*

We decided to use CodeGen to built that list of words, with each "bad" wrong being linked to its appropriate translation. We enlisted the help of our mentor to extract the embeddings outputted by CodeGen so that we could use them. We then used cosine similarity to match each keyword to its closest representation - its most similar keyword in the target language.

3 Results & Discussion

We ended up with an array of translated keywords that looked like this :

PYTHON:	**JAVA:**
abs	abstract
isinstance	instanceof
set	enum

We added this array to the VSCode extension that was previously hard-coded and tried it out. Our demo video, available on GitHub, shows that the extension consistently and reliably flagged and corrected keyword errors.

Given more time, we would have liked to train our own transcompiler model using Gensim and Word2Vec on Python. The obvious next step for this projects would be compilable code translation - not just keywords, which we would like to pursue in the future. In addition, we would add support for other popular IDE's and most importantly other languages.

Acknowledgements

We are grateful to Borealis AI for organizing this program. Many thanks to our mentor, Giuseppe Castiglione, for his invaluable help and advice.

References

[1] Baptiste Roziere, Marie-Anne Lachaux, Lowik Chanussot, and Guillaume Lample. Unsupervised translation of programming languages. *Advances in Neural Information Processing Systems*, 33, 2020.