Semantic Rule-based Automatic Code conversion System

***Abstract*—Most software employees are facing challenges on integrating programs written in different languages for imple- menting the techniques for their software development. This can be achieved by automating the conversion of programs using nat- ural language programming techniques. This research presents a novel ‘Semantic Rule-based Automatic Code conversion System (SRACS)’ that uses semantic layering, keyword identification, and a semantic rule-based constructor. The code snippets for ‘Hello World’, ’For Loop’, ‘While Loop’, ‘If else’, ‘Factorial’ and ‘Travelling Salesman Program’ are converted from Java to Python and vice versa, and the accuracies are presented. An average accuracy of 71.57% is achieved for the conversion of the code snippets from Java to Python, and a 77.07% is achieved for Python to Java. The accuracy is based on the ‘accuracy in the conversion of the variables’, ‘accuracy in the conversion of the attributes’ and in the ‘proper indentation of the code in the target code’.**

***Keywords*—Python, Java, code snippets, NLTK, semantic- analysis, translator, code conversion**

1. INTRODUCTION

The design phase and the development phase are the two parts of a Software Development Life Cycle. The design conversion using a programming language is an effortless task. The composition of a source code requires knowledge of algorithms in a specific area. There are two methods for re- writing the source code of a program: either by updating the existing code or by the creation of a new source code. Software engineering tools may be used for automating certain steps in the software development process [1].

Natural Language Processing (NLP) is at the peak of usage for Artificial Intelligence (AI) based systems that allows a computer to master the natural language. NLP can be helpful in broadening text processing beyond the range of simple syntactic processing to critical semantic processing, which is the natural ability of human beings. There are few diverse AI approaches that can be utilized by NLP applications to achieve the task of understanding the meaning intended of the human speech. A rule-based approach makes use of rules that use predefined conditions to map the syntax to a viable semantics. But a connectionist approach uses a learning strategy to develop a mapping algorithm. NLTK based Semantic Framework is helpful in the automated conversion of the pseudocode to the required learning based, computed, and

deducted baseline text [2]. In this research, a novel ‘Semantic Rule-based Automatic Code conversion System (SRACS)’ for conversion from Python to Java and vice versa is presented. The following sections include Background Research, Archi- tecture of SRACS, Simulation and Results, Conclusion and Future Work, followed by References.

1. BACKGROUND RESEARCH

Multiple processing layers in a deep learning neural net- work allows computational models to learn various levels of abstraction. Deep learning needs labelled datasets presented as ‘examples’ to train the computational modules prior to making use of it for an application. The training of such frameworks sometimes requires enormous amounts of data [3]. This will be helpful for the program to automatically discover the relevant correlations among input and output patterns. Natural Language Processing makes use of deep learning to extract the meaning of the desired patterns.

Text processing is a well-known approach for accomplishing the task of code conversion from one semantic format to another. In this methodology, a query table is maintained for mapping of the trained data. In practice, this approach is not attainable, as the conversion of a code from one semantic form to another requires more than just blind mapping of data [4]. The machine learning translation helps in the code conversion task using the inter-lingual representation to generate the target code. This process consists of two major steps: the inter- lingual conversion and generating the target code [5]. In the inter-lingual conversion, the pseudocode is converted into the linguistic form and then by using text processing the target code is generated. Compiler software is an example of this process where a low-level program is generated from a high- level program [6]. This approach has generated satisfactory formats with the help of Artificial Intelligence, but it was not able to generate target code from a pseudocode using natural

Language processing (NLP) [7].

Language Based Machine Translation (LBMT) is a method for machine translation that can be done either as lexeme- arranged or syntax situated [8]. In the lexeme-arranged methodology, the interpretation is accomplished based on the rule of lexical equality. In this, some of the units for interpretation is ‘words’ and ‘explicit’ that consists of specific phrases, but when it comes to LBMT, the general expression

of it would remain as in the original form. In a punctuation arranged methodology, the unit of interpretation describes the source text and is restricted to intra-sentential structures. Both methods give little knowledge into the utilization of the setting. The LBMT provides an interpretation utilizing the gathered instances of the past interpretation measures [9].

Deductive thinking is a methodology that is utilized to demonstrate the sufficiency of a hypothesis [10]. Inductive thinking is a methodology that begins with initial information and continues until an objective is reached [11]. These methodologies are the two different types of AI reasoning that are utilized to create the code from the planning phase. In light of these approaches a fully automated, deductive and inductive programming that can be utilized to produce portions of algorithms using UML charts and program fragments [12].

1. ARCHITECTURE OF SRACS

The architecture of the SRACS includes ‘Semantic Layer- ing’, ‘Keyword Identification’ and ‘Semantic Layer Construc- tor’ modules. Initially, the pseudo code is processed through a semantic layering technique that consists of two major steps, the segmentation and labelling, and the identification of the common units. In the segmentation and layering step, the entire pseudocode is segmented into single units and each unit is labelled as per the semantics. Then, the common units among the semantics are identified and grouped together and labelled. These are fed into a Natural Language Tool Kit (NLTK) that identifies all the general vocabularies. The architecture of the proposed method is shown in Fig.1.

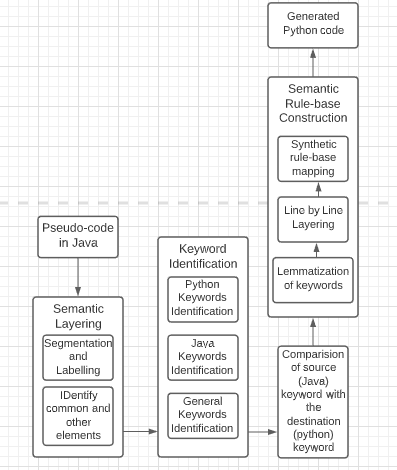


Fig. 1. Architecture for Semantic Rule-based Automatic Code conversion System (SRACS)

In order to perform Java to Python code conversion the key- words are identified using the ‘Java Keyword Identification’ module and compared with the ‘Python Keywords’ module

in the NLTK. The corresponding equivalent Python keywords are lemmatized and ‘line by line layering’ is done. In the lemmatization step the identified python keywords are grouped based on their similarity with respect to the Python vocabulary. In the ‘line by line layering’ step the lemmatized keywords are layered as per segmentation rules. Finally, the layered keywords are indented in the ‘Synthetic Code Formatting’ step to generate the python code.

1. SIMULATION AND RESULTS

The conversion of Java code to Python and vice versa are used for the SRACS. The pseudocode in Java for the ‘Hello World’ shown in Fig. 2(a) is given as input. This is translated through ‘Semantic Layering’ where all the attributes and the variables along with the notations are segmented based on their occurrence in the code format. These are labelled as shown in the Fig. 2(b). Once all the elements of the code are labelled, the common elements and all other elements are identified. This is done for the easy analysis of the code snippets, and also for the accurate mapping of the keywords.

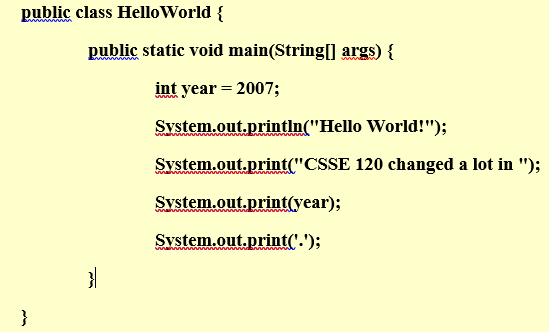


Fig. 2(a). Pseudo code for the ‘Hello world’ program

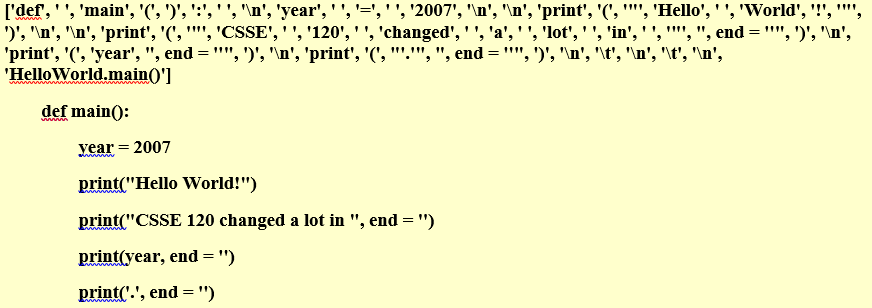


Fig. 2(b). Keyword identified for Pseudo code in Fig. 2(a)

The TensorFlow framework with the multi-layer neural network available in the NLTK package is trained with general vocabulary, Java code snippets and Python code snippets for ‘Keyword Identification’. This framework analyzes the labelled keywords of the pseudo code in Java and identifies the equivalent keywords in Python. These keywords are then re- arranged in an ordered format. Once all the equivalent keywords are identified, the construction of the Python code is done using the ‘Semantic rule-based Construction’ module. This includes ‘lemmatization’ of the code where all the identified keywords with similar code snippets are structured into a single line of code. Then the code is layered ‘line by line’. One of the important aspects of the Python code is that the indentation of the code needs to be observed. Otherwise,

there will be multiple errors occurring while compiling. Hence, this is done using the ‘Synthetic Rule-based Mapping’ module. The programs used for conversion using SRACS include the code snippets for ‘Hello World’, ‘for Loop’, ‘while Loop’, ‘If else’, ‘Factorial’, ‘Travelling salesman program’ and the ‘months with 31 days in a year’.

Python code is generated for the Java code in Fig. 2(a) and is shown below in the Fig.3

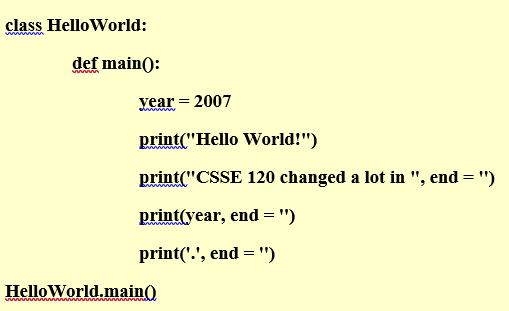


Fig. 3. Corresponding Python code for Java code in Fig. 2(a)

‘For-loop’ code in Java and the corresponding Python code are shown in Fig. 4(a) and 4(b) respectively using a graphical user interface.

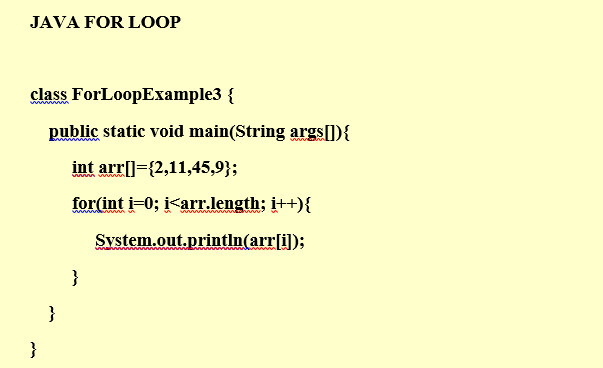


Fig. 4(a). ‘Java For loop’ conversion from Java to Python

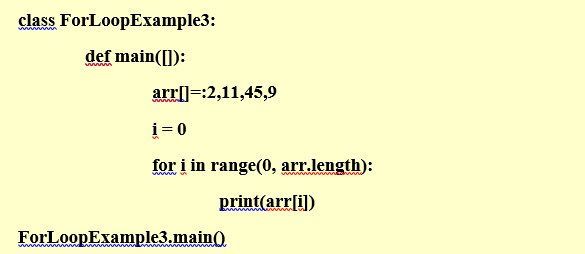


Fig. 4(b). ‘Python For loop’ conversion from Java to Python

‘if else’ code in Java and the corresponding Python code are shown in Fig. 5(a) and 5(b) respectively using a graphical user interface.

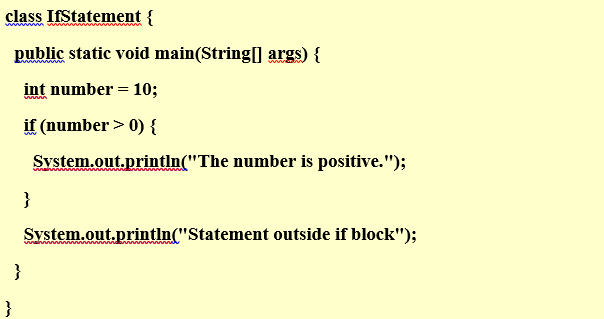


Fig. 5(a). ‘Java if else’ conversion from Java to Python

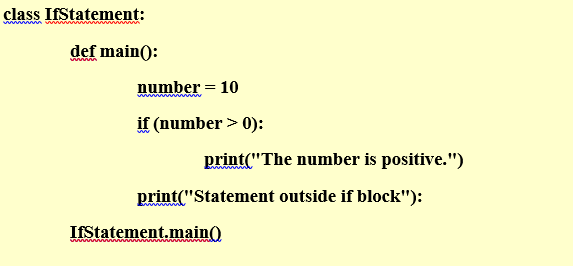


Fig. 5(b). ‘Python if else’ conversion from Java to Python

’While loop’ code in Java and the corresponding Python code are shown in Fig. 6(a) and 6(b) respectively using a graphical user interface.

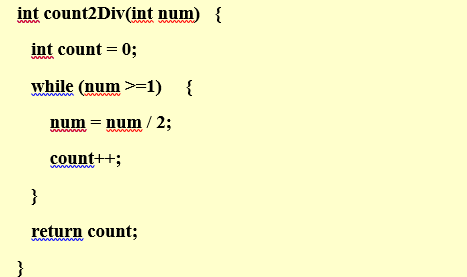


Fig. 6(a). ‘Java while loop’ conversion from Java to Python

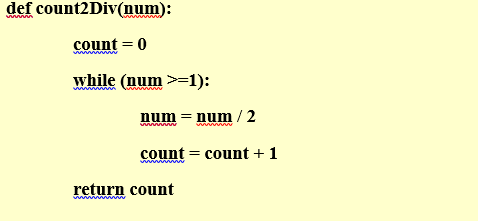


Fig. 6(b). ‘Python while loop’ conversion from Java to Python

‘Factorial’ code in Java and the corresponding Python code are shown in Fig. 7(a) and 7(b) respectively using a graphical user interface.

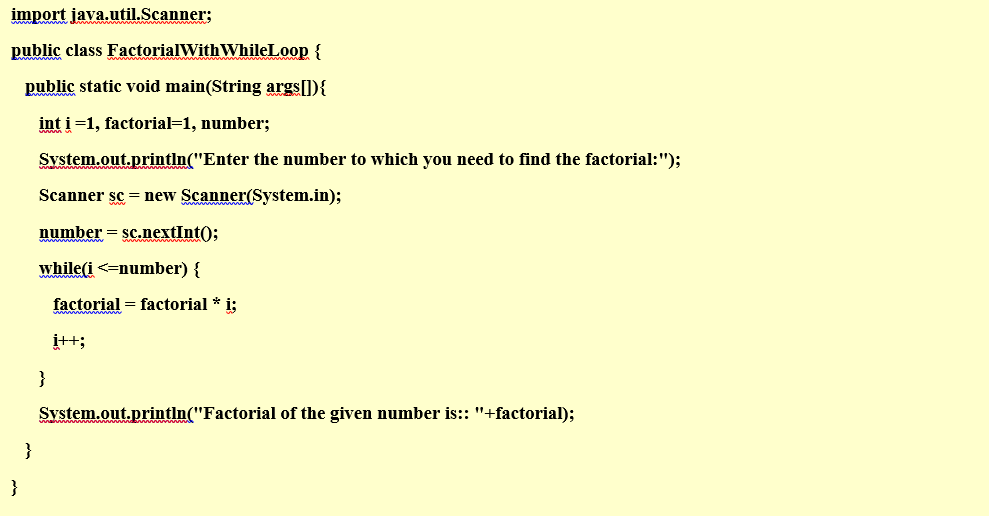


Fig. 7(a). ‘Java factorial program’ conversion from Java to Python

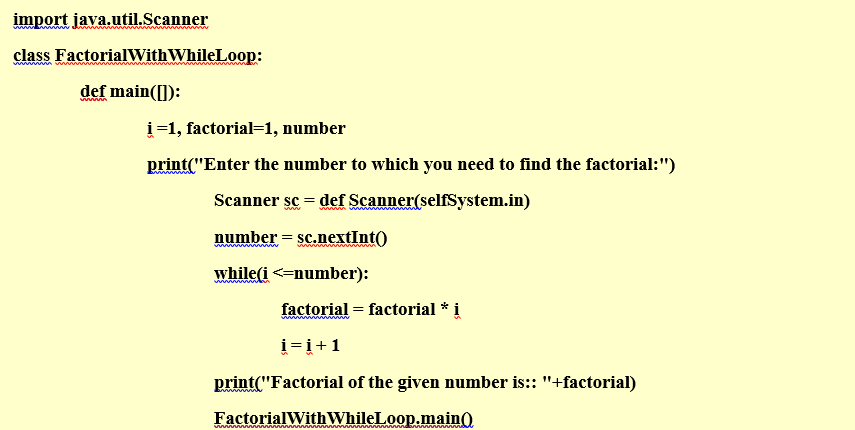


Fig. 7(b). ‘Python factorial program’ conversion from Java to Python

‘Program for months with 31 days’ code in Java and the corresponding Python code are shown in Fig. 8(a) and 8(b) respectively using a graphical user interface.

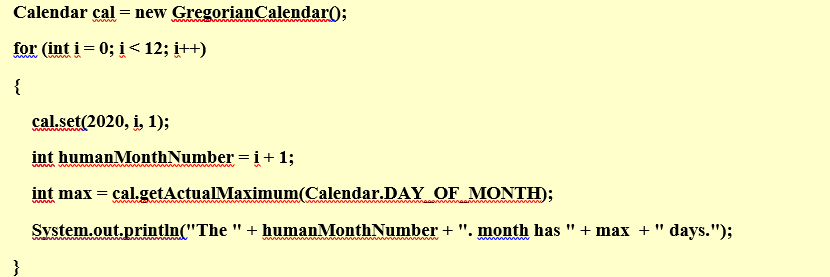


Fig. 8(a). ‘Java program for months with 31 days’ conversion from Java to Python

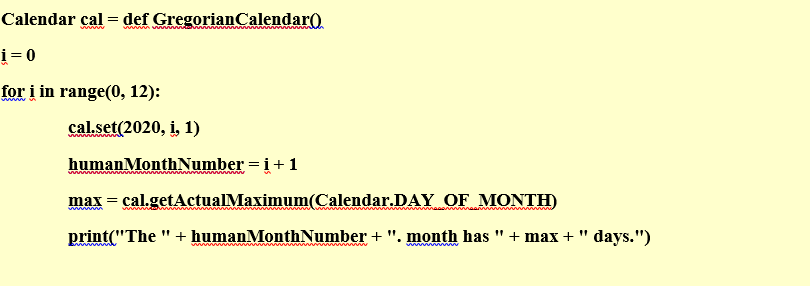


Fig. 8(b). ‘Python program for months with 31 days’ conversion from Java to Python

The results of the code conversions are assessed based on 3 factors from the source code snippets and the corresponding target code: ‘Accuracy in the conversion of the variables’, ‘Accuracy in the conversion of the attributes’ and in the ‘Proper indentation of the code in the target code’.

The results for conversion from Java to Python are given in Table 1.

Table 1. Results for Conversion from Java to Python

|  |  |  |  |
| --- | --- | --- | --- |
| Snippets used from Programs | Actual  code Snippets | Snippets  identified using  SRACS | Accuracy  (%) |
| ’Hello World’  ’For Loop’ ’if else’ ’While Loop’  ’Factorial’ ’Travelling  salesman’ | 24  35  47  43  65  78 | 20  23  33  35  47  55 | 83.3  65.7  70.2  74.33  72.3  70.25 |

The results for conversion from Python to Java are given in Table 2.

Table 2. Results for Conversion from Python to Java

|  |  |  |  |
| --- | --- | --- | --- |
| Snippets used from Programs | Actual code Snippets | Snippets identified using  SRACS | Accuracy  (%) |
| ’Hello World’ ’For Loop’  ’if else’ ’While Loop’ ’Factorial’  ’Travelling salesman’ | 18  46  53  45  79  83 | 14  32  39  31  61  65 | 77.7  69.65  73.5  68.8  77.2  78.3 |

The conversion module made for the ‘for Loop’ code achieved an accuracy of 65.7%. The code snippets obtained after the conversion lacked variable incrimination and function declaration, due to which 100% accuracy is not attained.

The conversion module made for the ‘while Loop’ code achieved an accuracy of 74.33%. The code snippets obtained after the conversion lacked the argument declaration, due to which 100% accuracy is not attained.

The conversion module made for the ‘if else’ code achieved an accuracy of 70.22%. The code snippets obtained after the conversion lacked the proper indentation, due to which 100% accuracy is not attained. An average accuracy of 71.57% is achieved for the conversion of the code snippets from Java to Python and 77.07% is achieved for Python to Java.

1. CONCLUSION AND FUTURE WORK

In this research, a novel ‘Semantic Rule-based Automatic Code Conversion System (SRACS)’ is presented that uses semantic layering, keyword identification, and a semantic rule- based constructor. In the traditional methods labeling of elements is done using the one-to-one element mapping tech- nique. In the SRACS, the mapping of the code is done using one-to-many mapping that helps in identifying the similar code snippets, and also in the segmentation and layering of the code. The TensorFlow framework available in the NLTK package is used for identification of keywords in the source codes. The accuracy of the converted code snippets from Java to Python is

71.57%. Also, the accuracy for Python to Java is 77.07%. The SRACS may be extended by training large sets of variables, attributes, and any other needs of the converted code snippets for any other programming languages. It may find application in the code conversion algorithms for COBOL to Java that Banks, Internal Revenue Services or Legacy Systems of industries use.

1. G. F. Luger, Artificial intelligence: structures and strategies for complex problem solving, 6th ed., Pearson, London, UK, 2008.
2. Brown, David. (2005). Artificial intelligence for design process improve- ment.
3. Danilchenko, Yuri and Fox, Richard. (2012). Automated Code Genera- tion Using Case-Based Reasoning, Routine Design and Template-Based Programming. 10.13140/2.1.1470.4641.

REFERENCES

1. Dan Jurafsky and James Martin. Speech and language processing. Pearson, 2014.
2. G. F. Luger, Artificial intelligence: structures and strategies for complex problem solving, 5th ed, 2007.
3. LeCun, Yann and Bengio, Y. and Hinton, Geoffrey. (2015). Deep Learning. Nature. 521. 436-44. 10.1038/nature14539.
4. D. Jurafsky and J. H. Martin, Speech and language processing, An In- troduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2014.
5. Nirenburg, S., and Mahesh, K. (1997). Knowledge-Based Systems for Natural Language Processing. The Computer Science and Engineering Handbook.
6. M. R. Amal, C. V. Jamsheedh and L. S. Mathew, Pseudocode to source programming language translator, 2016
7. Dirgahayu, Teduh and Nurul Huda, Sheila and Zukhri, Zainudin and Ratnasari, Chanifah. (2017). Automatic translation from pseudocode to source code: A conceptual-metamodel approach. 122-128. 10.1109/CY- BERNETICSCOM.2017.8311696.
8. Bonnie J. Dorr. 1997. Large-scale acquisition of LCS-based lexicons for foreign language tutoring. In Proceedings of the fifth conference on Applied natural language processing (ANLC ’97). Association for Computational Linguistics, USA, 139–146.
9. D. Jurafsky and J. H. Martin, Speech and language processing, vol. 3, Pearson London, London, 2014.