International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 01 | Jan 2023 www.irjet.net p-ISSN: 2395-0072

Recent Trends in Translation of Programming Languages using NLP Approaches

Dr. P. Venkateswara Rao[1], B. Sunil Kumar[2], Ch. Mourya[3], M. Akhil Reddy[4], P.V. Siddharth[5], Y. Jeevan Reddy[6]

[1] *Assistant Professor, Dept. of Computer Science and Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India*

[2,3,4,5,6] *Student, Dept. of Computer Science and Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India*

Abstract - *Java, C, C++ are a few of the most robust and popular programming languages still used today. Python, on the other hand, is a computer programming language that is often used to design websites, automate day-to-day IT tasks, and conduct data analysis. C/C++ and Java follow an imperative and structural programming model having strict semantic rules and functions whereas Python is an Object-based language that also provides a rich set of in-built libraries. Python is more readable and simpler. In this project, "Language portability to Python" we would like to implement a translator engine that translates multiple programming languages namely Java, C, C++ into Python code without changing the actual meaning and functionality of the original code. This results in an automated translation rather than having to rewrite the complete Python programme from start.*

***Key Words*: Translator, Compiler, Interpreter, Python, C, C++, Java.**

1. INTRODUCTION

Programming Languages Conversion has been a challenging topic for almost a decade. Converting a piece of code from one language to another entails more than just changing the syntax between the two languages. It involves executing transformation while trying to keep the structure and optimization intact.

Python has advanced in recent years to rank among the programming languages that are most often used globally. It's a programming language that's frequently employed for data analysis, software development, and process automation. Python is a versatile programming language that may be used to construct a broad range of applications and does not concentrate on a particular problem. Because of its versatility and beginner-friendliness, it has gone to the highest spot on the programming language list now in use.

Consider a scenario in which a developer or programmer experienced in structured programming languages like Java, C or C++ must implement a Python application without altering its intended use. Another scenario is a beginner programmer who wants to quickly learn Python but only has experience with C, C++, or Java.

A language translator, which translates one computer language to another with a single click, is useful for resolving such issues. Python and C/C++/Java are relative newcomers to the world of programming, but they have both earned a spot among the most widely used ones right now. Both of them have numerous strong characteristics that programmers want. Python is less difficult to learn for beginning programmers than Java. Python is more flexible and less complicated than Java, so studying programming in Python as a first language will allow one to advance more quickly. In addition to being more user-friendly and robust, Python is also simpler to read, comprehend, and debug. It also has a more natural coding style. Because it is a programming language with dynamic typing as opposed to Java's static typing, it is also more productive. Python is reliable and utilized by many enormous companies, like Google.

2. RELATED WORK

Wasi Uddin Ahmad et al.,[1] presented a corpus made of nearly 8,475 programming problems along with their Java and Python-based solutions. They gathered the dataset from open-source repositories and online coding platforms which include CodeJam, GeeksForGeeks, and Leetcode. They trained through three different models namely the No training model, the training from the scratch model, and the pre-trained model. Out of which PLBART model performed well compared to others however it failed to convert import statements and type causing severe type mismatch.

Eman J. Coco et al., [2] provided a paradigm for converting Java code to Python. Two stages make up this process: the first stage involves converting Java code to an intermediate language, and the second stage involves translating the intermediate language code into Python. They chose XML as an interpreted language because it may be understood by two different programs that are incompatible with one another, and because XML data is saved as text, lowering the risk of data loss. The translator reads each character in the Java file in order to determine the type and individual parts of the statements. Each Java instruction is translated into the name of the XML tag that best describes it, which is determined by the type of Java instruction. The tag attributes record the contents of the Java statement. Before processing the tree nodes, which are made up of XML tags, the translator extracts the Document Object Model tree from the XML file. Each XML tag is translated into a corresponding Python instruction, with the tag name determining the statement's type. The tag attributes are used to extract the parts of the Python statement. The fundamental Java components may all be converted to Python using this paradigm. This model's drawback is that it is unable to translate Java code that contains OOPs and data structures.

Baptiste Roziere et al.,[3] suggested a transformer design consisting of an encoder and a decoder is used in a sequence-to-sequence model. A monolingual approach was used to represent all of the programming languages. Unsupervised machine translation initialization, language modeling, and back-translation approaches were utilized to train it. They used the previously trained XLM model to initialize the model's encoder and decoder. Only 3% of the original reference was translated when going from C++ to Java, despite the fact that 61% of them passed the unit tests.

Prof. Satish Kuchiwale et al., [4]By employing recurrent neural networks and sequence-to-sequence mapping, a pseudo code is transformed into a python code. For the mapping of sequences, it employs an end-to-end strategy. It is made up of a decoder and an encoder. The encoder transforms the input sequence into a context vector using multi-layer LSTM cells. The decoder then receives this context vector and uses deep LSTM cells to produce the target sequence. Only logical assertions stated in simple English can be parsed by the algorithm.

Dony George et al., [5]The language is first processed by the compiler, after which the program is parsed to determine its structure and scanned to gather additional information about variable names, function names, etc. The conversion software then transforms the code into an intermediary language file, after which the code is further processed by a Language Optimization tool. This is translated to target code by the compiler's de-conversion procedure. This is unable to translate code across two distinct platforms.

Dr. Safwan Omer Hasson et al., [6]Pseudocode uses statements to express actions, and variable and function names are connected together by underscores. Summations and Counters must be initialized to zero before being trained using a neural network by defining a matrix with binary integers, creating random weights after initialization, and contrasting the neural network's actual output with the desired output. The output created does not match the target output due to the high error rate.

Wim T.L.P. Lavrijsen et al., [7]Cling, cffi, and PyPy's toolbox are combined in cppyy, a new module for PyPy-C that offers high-performance Python bindings for contemporary C++. Modern C++ makes it easier for interfaces to communicate purpose, which considerably helps automatic bindings generators. For instance, by making ownership and thread safety crystal clear. With size and distribution in mind, the cppyy module builds bindings in a lazy manner and has minimal dependence on the Python interpreter. Deconstructing high-level conceptions to low-level interfaces allowed for optimizations. On the other hand, the PyPy optimizer is designed to operate with higher-level constructs.

Karan Aggarwal et al., [8] used a Python 2 code to Python 3 code by statistical machine translation. He achieved a high BLEU score by combining data from two earlier experiments. He also researched cross-project training as well as testing to identify mistakes and eliminate disparities with past situations. In order to achieve the goal of creating translation models that closely resemble natural languages themselves, he has given a thorough analysis on modelling programming languages as natural languages.

Tom Simonsen et al.,[9] invested most of their work into translating Python code to “standard” CPP. His primary goal was to get most of the basic language elements translated. He implemented the translation, and by doing so he named the translator “Python2C”. He added that by implementing the Windows CE GUI translation, the translation of code from Python to CPP can be done even more accurately. Since the translation engine has been created, he suggested that it just needs tweaking and a few upgrades to make it more powerful. He also highlighted that Python2C is capable of translating GUIs if someone takes the time to do this.

Ian J. Davis et al., [10] recommended bash2py, a source-to-source translator for bash scripts that turns them into Python code. Using both the inbuilt parser of Bash and open-source bash code, Bash2py processed any bash script. On the other hand, Bash2py reimplements variable expansion in the Bash language to produce Python code that is more accurate. Bash2py transforms the majority of Bash scripts to Python, however, it requires human interaction to handle constructions that cannot be simply translated by the computer. He conducted experiments with the real-world Bash scripts from the open source. The bash2py was able to successfully translate 90 percent of the code.

Richard C Waters et al.,[11] suggested a two-step translation that uses abstraction and reimplementation. The abstraction stage analyzes the source program globally and attempts to comprehend the algorithm that is being used there. The reimplementation stage attempts to write a program in the needed target language using the abstraction's description as a starting point. This approach is more difficult and has a problem with incompleteness.

Xinyun Chen et al.,[12] proposed using an encoder-decoder framework with the tree-to-tree neural network in which the source tree is encoded into an embedding and the embedding gets decoded into the target tree. The encoder makes use of a Tree-LSTM to determine to embed for both the entire initial tree and subtree. The destination tree is then generated by the decoder starting at the root node. For increased efficiency, binary trees are created from the source tree and the target tree. It extends each node recursively while maintaining a queue of all nodes that need to be extended. Programs that are bigger than the taught ones are challenging.

Onkar Apte et al.,[13] proposed a model that translates the code written in C language to Python. They used Python language to design the translator. The facilities that Python offers and the ease of developing code are the key justifications for utilizing it. The work of constructing a translator was made simple by Python's abundance of built-in, ready-to-use methods suitable for string manipulation, such as split(). Without any problems and using the same logic, the translator can be written in any other language. There are actually three stages to the translating process. First, the ideal syntax is applied to the C code. The translator analyzes the file line by line in the second stage, identifying the line type that we have chosen at random. Each line is translated by using the line-type array established in the second step in the third phase, and the outcome is a fully translated Python code file. The model was able to translate variable declarations, comment lines, control statements, etc., The limitation of this model is it cannot translate structure and pointers present in C language.

Rahul Dubey et al., [14] proposed an Algorithm to code converter, also referred to as A2C converter. It is an interpreter or translation process that enables the user to just write a problem's algorithm in semi-natural English, which may then be translated into Java or C computer language code. This model consists of two phases, translator phase: employing POS Tagging, Classification by Naive Bayes Classifier, and Data Extraction to convert the input algorithm to an XML specification file. The main benefit of utilizing a classifier is that data extraction is made easier once the kind of statement is determined. Mapping phase: mapping the input algorithm's output XML specification file to the necessary C language code. The limitation of this model is it can only convert algorithms that are properly structured.

S. T. Gollapudi et al., [15] proposed a model that translates code based on semantics. It converts Java code into Python. The Natural Language ToolKit's Java Keyword Identification module is used to identify the keywords, and the matching Python Keyword Identification module is then used to compare the keywords to convert the identified keywords from Java to Python code. The following stage entails three modules that involve "line-by-line layering" and lemmatizing the comparable Python keywords. During the concordance stage, the discovered keywords of Python are arranged based on how much they resemble the vocabulary in Python.The "line-by-line layering" technique involves stacking keywords with lemmatization in accordance with segmentation rules. In the final phase, "Synthetic Code Formatting," the stacked keywords are indented to create the python code.

Neeta Verma et al. [16] proposed an NMT-based model. Neural machine translation (NMT) is a pragmatic method to machine translation that employs a large artificial neural network to model or produce the full phrase as a single integrated model in order to forecast or know the occurrence of a big sequence of words. a unidirectional, deep multi-layer recurrent neural network that uses the LSTM as a recurrent unit. The NMT model, at its most basic level, comprises two recurrent neural networks: the encoder RNN consumes the original words from the input without making any predictions; the decoder, on the other hand, predicts the following words while processing the target text.

Table -2.1: English to Global Language Translation

Related Work	Methodology	Evaluation Method	Outcome
Hector Llorens et al [25] (2010)	Conditional Random Fields probabilistic model (CRF), TIPSem	precision, recall, and Fβ=1 metrics	• For Spanish, TIPSem achieved the best Fβ=1 in all tasks. • For English, it obtained the best Fβ=1 in event recognition and classification, and event and document creation time links categorization.
Melvin Johnson, et.al (2017)	NMT and Multilingual model architecture	BLEU score metric	• train multilingual NMT models with a single model where all parameters are shared that can be used to translate between a number of different languages, • Without explicit bridging, zero-shot translation is proven to be feasible.
(Soft) alignment generated by the RNN search Encoder – Decoders Model	(Soft) alignment generated by the RNN search Encoder–Decoders Model	Alignment, accuracy	• outperforms the conventional RNNencdec • produced translation performance on par with phrase-based statistical machine translation already in use.
Philipp Koehn et al (2017)	MT and SMT	probability mass, BLEU scores	• When put in situations that are significantly different from training settings, neural translation models do not behave robustly. • There are still many obstacles for neural machine translation to get past, mostly performed outside of the target language and when resources are limited.

Table -2.2: Global to Local Language Translation

Zelated Work	Methodology	Evaluation Method	Outcome
Effective preprocessing based neural machine translation for English to Telugu cross-language information retrieval (2022)	Using RNN and LSTM machine learning models.	Accuracy, Perplexity, Cross-entropy and BLUE scores	• BLEU score of RNN model with and without replication were 46.03 and 46.87 respectively. • BLEU score of LSTM model with and without replication were 46.38 and 47.19 respectively.
Machine Translation System Using Deep Learning for English to Urdu (2022)	LSTM-based deep learning encoder-decoder model.	BLUE, F-measure, NIST, WER.	• The proposed system after extensive simulations achieves an average BLEU score of 45.83
Rule based Sentence Simplification for English to Tamil Machine Translation System (2020)	Rule based technique.	Accuracy	• 200 sentences are given to the rule based English to Tamil machine translation system out of which 140 sentences are incorrect because of syntax and reordering error.
English to Bengali Multimodal Neural Machine Translation using Transliteration-based Phrase Pairs Augmentation (2022)	Transliteration based phrase pairs augmentation approach.	BLUE and RIBES	• Text-only NMT Evaluation BLEU and RIBES scores were 40.9 and 0.75 respectively. • Multi modal NMT Evaluation BLEU and RIBES scores were 43.9 and 0.78 respectively.

Table -2.3: Local to Local Language Translation

Related Work	Methodology	Evaluation Method	Outcome
Sivaji Bandyopadhyay (2020)	Stemming and Zonal Indexing	effectiveness	● A robust stemmer is required for the highly inflective Indian languages. ● Machine-readable bilingual dictionaries with more coverage have improved the results.
Vandan Mujadia (2022)	Parallel Corpora	Average sentence length	● Iterative Back-translation driven Post-Editing can be used for similar parallel corpora creation work. ● Carefully created and curated parallel corpora boost the translation performance even with the lower parallel corpora size
Jagadeesh Jagarlamudi(2019)	Machine Translation Systems	threshold	● on CLEF data set, a Hindi to English cross-lingual information retrieval system using a simple word by word translation of the query with the help of a word alignment table was able to achieve ~ 76% of the performance of the monolingual system ● word translations with no threshold on the translation probability gave the best results
Mallamma V. Reddy (2020)	Mapping	Word Precision	● English has Subject Verb Object (SVO) structure while Kannada has Subject Object Verb (SOV) structure in Machine translation will be unraveled by using morphology ● we can also use language identification module for translation with the help of bilingual dictionary

Table -2.4: Programming Language Translation

Related Work	Methodology	Evaluation Method	Outcome
Jalil Nourisa (2021)	Agent based modeling (ABM)	Scalability, Versatility	● The result shows that the ABM is an effective method for translating from c++ to python. ● Provides several built-in functions which are demanded for the translation.
Wasi Uddin Ahmad (2022)	PLBART	Accuracy	● The models perform relatively well in terms of the lexical match. ● Aimed to improve the converting of import statements.
Baptiste Roziere (2020)	Sequence to Sequence	Correctness, Precision	● The model requires no expertise in the source or target languages. ● This method relies exclusively on monolingual source code.
Geir Yngve Paulsen (2020)	Armadillo Seismic Lab	Complexity, Accuracy	● It handles realistic matlab codes for translating into other programming languages. ● This model improves the optimization of the code that has been written.

3. PROPOSED METHODOLOGY

Our proposed software is hosted on the web through the Django framework providing a neat user interface where a user can initially choose a programming language from C/C++ or Java as the source code. By selecting the language, the user can now paste the code that needs to be translated into the text editor. On clicking the translate button the entire code is sent to the application logic analyzer.

The primary objective of the logic analyzer is to process line-by-line source code to equivalent Python code without changing the functionality or purpose of the code. A syntax tree is generated from the structured code which is further processed using NLP tools. Finally, the generated code is optimized using the APIs and served to the user.

C++

```
#include<iostream>
void main(){
int a;
cin>>a;
cout<<"Value of a: "<<a;
}
```

Python

```
a = int(input())
print("Value of a:",a)
```

Fig -3.1: C++ to Python

C++

```
#include<iostream>
void main(){
int a=10,b=20,c;
c = a + b;
cout<<"Sum = "<<c;
}
```

Python

```
a,b,c = 10,20,0
c = a + b
print("Sum =",c)
```

Fig -3.2: C++ to Python

C++

```
#include<iostream>
int sum(int x, int y){
return x + y;
}
void main(){
int a=10, b=20;
int c = sum(a, b);
cout<<"Sum = "<<c;
}
```

Python

```
def sum(x, y):
    return x + y
a,b = 10,20
c = sum(a,b)
print("Sum =", c)
```

Fig -3.3: C++ to Python

4. CONCLUSIONS

We conducted extensive research for this paper on Language Portability to Python, looking at a wide range of research publications on Programming Language interconversion. We learned that most of the models were only able to convert basic elements of code. With current technologies, the conversion of codes containing structures, pointers, data structures, and OOPs is not possible.

Additionally, we have learned about several proposed and existing systems through research publications, which has helped us develop a new model that would make translation much more efficient.

ACKNOWLEDGEMENT

Special thanks to our team guide, Mr. P. Venkateswara Rao, for all of his support and direction, which helped the literature survey portion of the project be successfully completed and yield positive results at the end.

REFERENCES

[1] Wasi Uddin Ahmad , Md Golam Rahman Tushar,Saikat Chakraborty‡ , Kai-Wei Chang.(2021). AVATAR: A Parallel Corpus for Java-Python Program Translation. arXiv:2108.11590v1.

[2] Coco, Eman & Osman, Hadeel & Osman, Niemah. (2018). JPT : A Simple Java-Python Translator. Computer Applications: An International Journal. 5. 01-18. 10.5121/caij.2018.5201.

[3] Marie-Anne Lachaux, Baptiste Roziere, Lowik Chanussot, Guillaume Lample. (2022). Unsupervised Translation of Programming Languages. arXiv: 2006.03511v3 [cs.CL]

[4] Vinay Patil, Rakesh Pawa2, Prasad Parab, Prof. Satish Kuchiwale. (2020). Pseudocode to Python Translation using Machine Learning. International Research Journal of Engineering and Technology (IRJET)

[5] Dony, George & Priyanka, Girase & Mahesh, Gupta & Prachi, Gupta & Aakanksha, Sharma. (2010). Programming Language Inter-conversion. International Journal of Computer Applications. 1. 10.5120/419-619.

[6] Dr. Safwan Omer Hasson, Fatima Mohammed Rafie.(2014). Automatic Pseudocode to Source Code Translation Using Neural Network Technique.

International Journal of Engineering and Innovative Technology (IJEIT)

[7] W. T. L. P. Lavrijsen and A. Dutta, "High-Performance Python-C++ Bindings with PyPy and Cling," 2016 6th Workshop on Python for High-Performance and Scientific Computing (PyHPC), 2016, pp. 27-35, doi: 10.1109/PyHPC.2016.008.

[8] Aggarwal K, Salameh M, Hindle A. 2015. Using machine translation for converting Python 2 to Python 3 code. PeerJ PrePrints 3:e1459v1 https://doi.org/10.7287/peerj.preprints.1459v

[9] Simonsen, Tom.(2015). Translating Python to C++ for palmtop software development. universitetet i oslo institutt for informatikk.

[10] I. J. Davis, M. Wexler, Cheng Zhang, R. C. Holt and T. Weber, "Bash2py: A bash to Python translator," 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), 2015, pp. 508-511, doi: 10.1109/SANER.2015.7081866.

[11] R. C. Waters, "Program translation via abstraction and reimplementation," in IEEE Transactions on Software Engineering, vol. 14, no. 8, pp. 1207-1228, Aug. 1988, doi: 10.1109/32.7629.

[12] Xinyun Chen, Chang Liu, Dawn Song.(2018). Tree-to-tree Neural Networks for Program Translation, UC Berkeley

[13] Apte, Onkar & Agre, Shubham & Adepu, Rajendraprasad & Ganjapurkar, Mandar. (2021). C TO PYTHON PROGRAMMING LANGUAGE TRANSLATOR. 10.1729/Journal.26932.

[14] R. Dubey, B. Edward, R. Lewis and P. Karunakaran, "Algorithm to Code Converter," 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), 2018, pp. 657-661, doi: 10.1109/ICOEI.2018.8553950.

[15] S. T. Gollapudi and S. Sasi, "Semantic Rule-based Automatic Code conversion System," 2020 International Conference on Data Science and Engineering (ICDSE), 2020, pp. 1-5, doi: 10.1109/ICDSE50459.2020.9310169

[16] Ms. Neeta Verma, Abhay Jain, Animesh Basak, Kshitij Bharti Saksena.(2018). Survey and Analysis on Language Translator Using Neural Machine Translation. International Research Journal of Engineering and Technology (IRJET)

[17] Llorens, H., Saquete, E., & Navarro, B. (2010, July). Tipsem (english and spanish): Evaluating crfs and semantic roles in tempeval-2. In Proceedings of the 5th International Workshop on Semantic Evaluation (pp. 284-291)

[18] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. arXiv:1611.04558v2 [cs.CL] 21 Aug 2017

[19] Dzmitry Bahdanau,KyungHyun Cho,Yoshua Bengio. NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. arXiv:1409.0473v7 [cs.CL] 19 May 2016

[20] Philipp Koehn, Rebecca Knowles, Six Challenges for Neural Machine Translation, arXiv:1706.03872v1 [cs.CL] 12 Jun 2017

[21] B. N. V. Narasimha Raju; M. S. V. S. Bhadri Raju; Satyanarayana, K V V. IAES International Journal of Artificial Intelligence; Yogyakarta Vol. 10, Iss. 2, (Jun 2021): 306-315.

[22] Syed Abdul Basit Andrabi, Abdul Wahid, "Machine Translation System Using Deep Learning for English to Urdu", Computational Intelligence and Neuroscience, vol. 2022, Article ID 7873012, 11 pages, 2022

[23] Poornima, C., et al. "Rule based sentence simplification for English to Tamil machine translation system." International Journal of Computer Applications 25.8

[24] Sahinur Rahman Laskar, Pankaj Dadure, Riyanka Manna, Partha Pakray, and Sivaji Bandyopadhyay. 2022.

[25] Bandyopadhyay, S., Mondal, T., Naskar, S. K., Ekbal, A., Haque, R., & Godhavarthy, S. R. (n.d.). Bengali, Hindi and Telugu to English Ad-Hoc Bilingual Task at CLEF 2007. Advances in Multilingual and Multimodal Information Retrieval, 88–94.

[26] Vandan Mujadia and Dipti Sharma. 2022. The LTRC Hindi-Telugu Parallel Corpus. In Proceedings of the Thirteenth Language Resources and Evaluation Conference, pages 3417–3424, Marseille, France. European Language Resources Association.

[27] Jagarlamudi, J., & Kumaran, A. (n.d.). Cross-Lingual Information Retrieval System for Indian Languages. Advances in Multilingual and Multimodal Information Retrieval, 80–87.

[28] Reddy, M. V., & Hanumanthappa, M. Indic Language Machine Translation Tool: English to Kannada/Telugu. Multimedia Processing, Communication and Computing Applications, 35–49.

[29] George, Dony, Priyanka Girase, Mahesh Gupta, Prachi Gupta, and Aakanksha Sharma. "Programming language inter-conversion." International Journal of Computer Applications 1, no. 20 (2010): 68-74.

[30] Khoirom, S., Sonia, M., Laikhuram, B., Laishram, J., & Singh, T. D. (2020). Comparative analysis of Python and Java for beginners. Int. Res. J. Eng. Technol, 7(8), 4384-4407.

[31] Rai, S., & Gupta, A. (2019). Generation of pseudo code from the python source code using rule-based machine translation. arXiv preprint arXiv:1906.06117

[32] Bonthu, S., Sree, S. R., & Krishna Prasad, M. H. M. (2021, August). Text2PyCode: Machine Translation of Natural Language Intent to Python Source Code. In International Cross-Domain Conference for Machine Learning and Knowledge Extraction (pp. 51-60). Springer, Cham.