JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Language Portability to Python

[1]Dr.P.Venkateswara Rao, [2]P.V.Siddharth, [3]B.Sunil Kumar, [4]Ch.Mourya, [5]Y.Jeevan Reddy, [6]M.Akhil
[1]Associate Professor,[2, 3, 4, 5, 6] Student
[1, 2, 3, 4, 5, 6]Dept. of Computer Science and Engineering,
[1, 2, 3, 4, 5, 6]VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India

Abstract : C, C++ and Java are popular programming languages that are widely used for developing complex and high-performance software systems. They are known for their robustness and efficiency in handling system-level tasks. These languages follow the imperative and structural programming paradigm, where programs are written as a sequence of instructions that are executed in a specific order. They have strict semantic rules and functions that ensure code correctness and minimize errors. Python, on the other hand, is an object-based programming language that provides a rich set of in-built libraries. It is often used for web development, automation of IT tasks, and data analysis. Python has gained popularity due to its simplicity, readability, and ease of use. Unlike C, C++, and Java, Python follows an object-oriented programming paradigm that focuses on objects and classes. It supports dynamic typing, which means that the data type of a variable is determined at runtime.

In the project "Language portability to Python," the aim is to implement a translator engine that can convert code written in C, C++, and Java into Python without altering the original functionality of the source code. This would enable the automatic translation of code written in these languages into Python, eliminating the need to write the entire Python program from scratch. The translated code would be able to run on any platform that supports Python. Implementing such a translator engine requires a deep understanding of the syntax and semantics of the source languages and the ability to map them to their Python equivalents. It involves parsing the input code, analyzing the syntax tree, and generating Python code that performs the same operations. The process requires a robust translation engine that can handle complex code structures and maintain code correctness and functionality.

IndexTerms **- Translator, Compiler, Interpreter, Python, C, C++, Java.**

I. INTRODUCTION

Programming Languages Conversion has been a challenging topic for almost a decade. Converting a piece of code from one language to other 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.

II. LITERATURE SURVEY

The complete literature survey for the project was published as a survey paper [16], a part of the literature is included here.

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, even though 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.

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	• TIPSem obtained the highest F=1 in all challenges for Spanish. • It achieved the best F=1 in event identification and classification, as well as document and event creation time linkages categorization, for English.
Melvin Johnson et al [26] (2017)	NMT and Multilingual model architecture	BLEU score metric	• Train multilingual NMT models with a single model that shares all parameters and may be used to translate between many languages. • Zero-shot translation has been demonstrated to be viable in the absence of explicit bridging.
Dzmitry Bahdanau et al [27] (2016)	(Soft) alignment generated by the RNN search Encoder– Decoders Model	Alignment, accuracy	• The conventional RNN encdec is outperformed.. • Achieved translation performance which is as effective as existing phrase-based statistical machine translation
Philipp Koehn et al [28] (2017)	MT and SMT	probability mass, BLEU scores	• Neural translation models do not operate robustly when placed in scenarios that are considerably different from training conditions. • There are still numerous challenges for neural machine translation to overcome, most of which are conducted outside of the target language and with limited resources.

Table 2.2 Global to Local Language Translation:

Related Work	Methodology	Evaluation Method	Outcome
B. N. V. Narasimha Raju[29] (2022)	Using RNN and LSTM machine learning models.	Accuracy, Perplexity, Cross-entropy, and BLUE scores.	• BLEU scores of RNN models with and without replication were 46.03 and 46.87 respectively. • LSTM models with and without replication had BLEU scores of 46.38 and 47.19, respectively.
Syed Abdul Basit Andrabi[30] (2022)	LSTM-based deep learning encoder-decoder model.	BLUE, F-measure, NIST, WER.	• Following extensive simulations, the suggested system gets an average BLEU score of 45.83.
Poornima, C[31] (2020)	Rule-based technique.	Accuracy	• The English-to-Tamil translation system machine based on rules is given 200 phrases, 140 of which are inaccurate due to syntax and reordering errors.
Sahinur Rahman Laskar[32] (2017)	Transliteration based phrase pairs augmentation approach.	BLUE and RIBES	• The BLEU and RIBES scores for text-only NMT Evaluation were 40.9 and 0.75, respectively. • The RIBES and BLEU scores for the Multi-modal NMT Evaluation were 43.9 and 0.78, respectively.

Table 2.3 Local to Local Language Translation:

Related Work	Methodology	Evaluation Method	Outcome
Sivaji Bandyopadhyay[33] (2020)	Stemming and Zonal Indexing	Effectiveness	• For the highly inflected Indian languages, a strong stemmer is necessary. • More coverage in machine-readable bilingual dictionaries has improved the findings.
Vandan Mujadia[34] (2022)	Parallel Corpora	Average sentence length	• Back-translation driven by iteration Similar parallel corpora development activity may be done with post-editing. • Parallel corpora that have been carefully produced and vetted improve translation speed even when the parallel corpus size is small.
Jagadeesh Jagarlamudi[35] (2019)	Machine Translation Systems	Threshold	• A cross-lingual information retrieval system from Hindi to English that used a basic A term by term translation of the request using a word alignment table achieved 76% of the monolingual system performance. • The best results came from word translations with no translation probability threshold..
Mallamma V. Reddy[36] (2020)	Mapping	Word Precision	• In machine translation, English has a Subject Verb Object (SVO) structure, whereas Kannada has a Subject Verb Object (SOV) structure. • A language identification module, in conjunction with a multilingual dictionary, can also be used for translation.

Table 2.4 Programming Language Translation:

Related Work	Methodology	Evaluation Method	Outcome
Jalil Nourisa[37] (2021)	Agent-based modeling(ABM)	Scalability, Versatility	• The results demonstrate that the ABM is a viable approach for converting from C++ to Python. • Supports multiple built-in functions required for translation.
Wasi Uddin Ahmad[38] (2022)	PLBART	Accuracy	• In terms of lexical matching, the models perform reasonably well. • The goal was to improve the conversion of import statements.
Baptiste Roziere[39] (2020)	Sequence to Sequence	Correctness, Precision	• The model does not need any knowledge of the source or destination languages. • This technique is entirely based on single source code.
Geir Yngve Paulsen[40] (2020)	Armadillo Seismic Lab	Complexity, Accuracy	• It works with actual Matlab code for translation into other languages. • This model increases the optimization of previously written programs.

III. PROPOSED METHODOLOGY

Our 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.

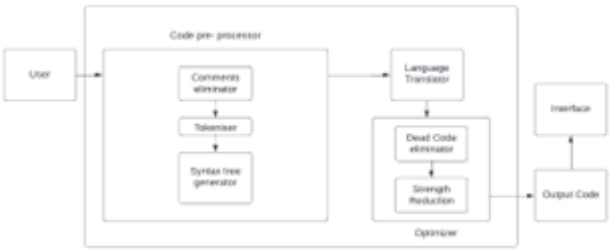

Fig 3.1: System Architecture

The entire application is divided into three parts - Preprocessor, Translator, and Optimiser module. The main task of the preprocessor is to parse the input code and generate a syntax tree. Then the tree is visited node by node and translated statement by statement to Python. The translator uses Natural Language Processing tools, and string manipulation to achieve this. The optimization of the output Python code is done using AST and ASTOR modules where the dead code is eliminated.

IV. IMPLEMENTATION

The implementation of the language-translation consists of various steps. The entire application is mainly divided into three modules namely the pre-processor module, the translator module, and the code optimizer module. The entire architecture is structured by the Django framework.

Preprocessing:

The primary function of the preprocessing module is to analyze the input code thoroughly, thoroughly examine each component, and meticulously convert it into a syntax tree, which is a structured representation of the syntactic structure of the code, capturing the hierarchical relationships between its different elements. This meticulous generation of syntax trees is crucial as it serves as a fundamental step in comprehending and interpreting the intricate structure of the program that is intended to be translated into the Python language, ensuring accurate and reliable translation.

Translation:

The translator uses NLP (Natural Language Processing) and string manipulation to identify and convert the code. For instance, to convert a printf statement in C language to the equivalent print statement in Python.

Optimization:

Code optimization is important as it can lead to improved performance, reduced resource usage, enhanced user experience, cost savings, future-proofing, better code quality, and competitive advantage. It is a critical aspect of software development that should be considered to ensure efficient and effective software or system operation. The below function uses “ast” and “astor” Python modules to eliminate the dead code from the output Python code.

User Interface:

The user interface is developed using django and its templates. The following HTML code provides the user with an editing text field where they can upload the input code to the translator.

Fig 4.1: User Interface

V. RESULTS

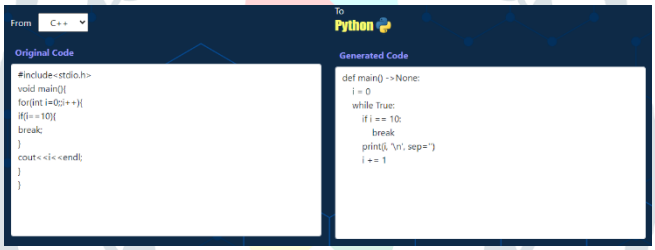

Fig 5.1: Loop Statements

The above image shows the conversion of the nested condition statement inside the loop statement.

Fig 5.2: STL

The above image shows the conversion of “Vector” container in C++ to equivalent “List” in Python Language.

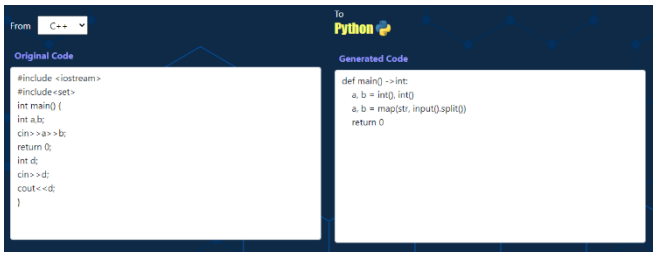

Fig 5.3: Optimization

The above code shows that the statements following the return statement are eliminated since they have no purpose in the overall output of the code.

Input file	LOC	Output file	Accuracy (10(	Code Quality score (10)	Error Rate (Tru	Memory Consumption (MB)
input1.c	15	output1.py	100	4.17	No	0.01
input2.c	20	output2.py	100	-10	No	0.008
input3.c	15	output3.py	10.2	ERROR	Yes	
input4.c	35	output4.py	100	9.06	No	0.04
input5.c	30	output5.py	75.1	-6.67	No	0.02
input6.c	50	output6.py	74.5	-8.3	No	0.03
input7.c	50	output7.py	5	ERROR	Yes	0.05
input8.c	65	output8.py	65.5	-7.97	No	0.03
input9.c	60	output9.py	70	-7.78	No	0.001
input10.c	70	output10.py	11	ERROR	Yes	
Average			**61.13**	**-3.927142857**	**0.3**	**0.023625**

An average accuracy score of 73.17% was secured by the translator. An error rate of 0.3 is recorded among the three programming languages using PYLINT. An average memory consumption of 0.023 MB was recorded across the translated Python code. Due to the optimizer, which is integrated into the translator, the performance improved significantly. Hence, increasing the efficiency of the generated code.

VI. CONCLUSION

This application offers users the ability to translate code from C/C++/Java to Python with a single click. This new application boasts several benefits, including improved performance and optimized memory consumption compared to generic online translators. As the number of lines of code increases, the benefits of using this new application become even more evident. Its ability to produce more efficient code with better memory management is particularly beneficial for larger projects. Despite the fact that the scope of the language is limited to the specific languages mentioned, the performance of the translator remains accurate. This means that users can expect to receive accurate and reliable translations every time they use the application.

Overall, this new development offers a significant improvement over generic online translators and represents a major advancement in the field of coding. Its ability to simplify the translation process and produce more efficient code is sure to be a valuable asset for developers everywhere.

VII. ACKNOWLEDGMENT

Special thanks to our team guide, Dr. 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] Dr. P. Venkateswara Rao, B. Sunil Kumar, Ch. Mourya, M. Akhil Reddy, P.V. Siddharth, Y. Jeevan Reddy, "Generalized deep learning model for Classification of Gastric, Colon and Renal Cancer", International Research Journal of Engineering and Technology (IRJET), Volume: 10 Issue: 01, e-ISSN: 2395-0056, p-ISSN: 2395-0072.

[17] G. Y. Paulsen, J. Feinberg, X. Cai, B. Nordmoen and H. P. Dahle, "Matlab2cpp: A Matlab-to-C++ code translator," 2016 11th System of Systems Engineering Conference (SoSE), 2016, pp. 1-5, doi: 10.1109/SYSOSE.2016.7542966

[18] Baptiste Roziere, Marie-Anne Lachaux, Lowik Chanussot, Guillaume Lample, Deep learning to translate between programming languages,July 21, 2020

[19] Nourisa, J, Zeller-Plumhoff, B, Willumeit-Römer, R. CppyABM: An open-source agent-based modeling library to integrate C++ and Python. *Softw: Pract Exper*. 2022; 52(6): 1337– 1351. doi:10.1002/spe.3067

[20] 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)

[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. https://doi.org/10.1155/2022/7873012

[23] Poornima, C., et al. "Rule based sentence simplification for english to tamil machine translation system." International Journal of Computer Applications 25.8 (2011): 38-42.

[24] English to Bengali Multimodal Neural Machine Translation using Transliteration-based Phrase Pairs Augmentation](https://aclanthology.org/2022.wat-1.14) (Laskar et al., WAT 2022)..

[25] 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).

[26] 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

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

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

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.

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

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

[37] 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.

[38] 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.

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

[40] 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

[41] Cottom, T. L. (2003). Using SWIG to bind C++ to Python. *Computing in Science & Engineering*, *5*(2), 88-97.

[42] Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[43] Fischer, T., Hirn, D., & Grust, T. (2022, June). Snakes on a Plan: Compiling Python Functions into Plain SQL Queries. In *Proceedings of the 2022 International Conference on Management of Data* (pp. 2389-2392).

[44] Jurica, P., & Van Leeuwen, C. (2009). OMPC: an open-source MATLAB®-to-Python compiler. *Frontiers in NEUROINFORMATICS*, 5.

[45] Marcelino, M., & Leitão, A. M. (2022). Extending PyJL-Transpiling Python Libraries to Julia. In *11th Symposium on Languages, Applications and Technologies (SLATE 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[46] Villar, J. I., Juan, J., Bellido, M. J., Viejo, J., Guerrero, D., & Decaluwe, J. (2011, April). Python as a hardware description language: A case study. In *2011 VII Southern Conference on Programmable Logic (SPL)* (pp. 117-122). IEEE.

[47] Karpiński, A. M. (2022). *Automatic translation of programs source codes from Python to C# programming language* (Doctoral dissertation, Zakład Sztucznej Inteligencji i Metod Obliczeniowych).

[48] Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., & Nakamura, S. (2015, November). Learning to generate pseudo-code from source code using statistical machine translation. In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 574-584). IEEE.

[49] Skanda, V. C., Jayaram, R., & Kumar, N. S. (2021). Refactoring OOPerl and Translating it to Python. In *Innovative Data Communication Technologies and Application* (pp. 375-386). Springer, Singapore.

[50] Marcelino, M., & Leitão, A. M. (2022). Transpiling Python to Julia using PyJL.

[51] Cannon, B. (2017). Porting python 2 code to python 3.scicomp.ethz.ch

[52] Buddrus, F., & Schödel, J. (1998, February). Cappuccino—A C++ to Java translator. In *Proceedings of the 1998 ACM symposium on Applied Computing* (pp. 660-665).

[53] Banbara, M., Tamura, N., & Inoue, K. (2005, October). Prolog cafe: A prolog to java translator system. In *International Conference on Applications of Declarative Programming and Knowledge Management* (pp. 1-11). Springer, Berlin, Heidelberg.

[54] de Freitas, A. F. (2005). From Circus to Java: Implementation and verification of a translation strategy. *Master's thesis, University of York*.

[55] Ancona, D., Anderson, C., Damiani, F., Drossopoulou, S., Giannini, P., & Zucca, E. (2002). A type preserving translation of F ickle into Java. *Electronic Notes in Theoretical Computer Science*, *62*, 69-82.

[56] An, K., Meng, N., & Tilevich, E. (2018, May). Automatic inference of java-to-swift translation rules for porting mobile applications. In *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems* (pp. 180-190).

[57] T. Patten and P. Jacobs, "Natural-language processing," in IEEE Expert, vol. 9, no. 1, pp. 35-, Feb. 1994, doi: 10.1109/64.295134.

[58] M. Youssef, K. Zakaria, B. Jamal, B. Toumi and B. M. Gaouth, "Text to Code Conversion Using Deep Learning for NLP," 2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2020, pp. 1-5, doi: 10.1109/ISAECT50560.2020.9523647.