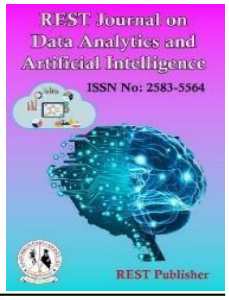
REST Journal on Data Analytics and Artificial Intelligence
Vol: 3(1), March 2024
REST Publisher; ISSN: 2583-5564
Website: http: //restpublisher.com/journals/jdaai/
DOI: https://doi.org/10.46632/jdaai/3/1/8



Analysis of Natural language processing for code generation by using COPRAS Method

Madhusudhan Dasari sreeramulu
Leading financial institution USA.
Corresponding author: *dsmadhu007@gmail.com*

Abstract: *Natural Language Processing has emerged as a powerful tool in various fields, including code generation. This paper explores the application of NLP techniques for the automatic generation of code from natural language specifications. The goal is to bridge the gap between human-readable requirements and machine-executable code, making software development more accessible and efficient. NLP for code generation serves as a bridge between human-readable natural language specifications and machine-executable code. This is particularly significant as it facilitates communication between developers and non-technical stakeholders, allowing them to contribute to the software development process without detailed programming knowledge. As the demand for software development continues to grow, there is a persistent talent gap in the industry. NLP for code generation can contribute to addressing this gap by allowing individuals with domain expertise to participate in software development without requiring extensive programming skills. The COPRAS-G method requires identifying selection criteria; evaluating information related to these criteria, and developing methods to evaluate Meeting the participant's needs Criteria for doing in order to assess the overall performance of the surrogate. Decision analysis involves a Decision Maker (DM) Situation to do consider a particular set of alternatives and select one among several alternatives, usually with conflicting criteria. For this reason, the developed complexity proportionality assessment (COPRAS) method can be used. From the result TRANX is got the first rank whereas is the Tree2 tree is having the lowest rank.*
Keywords: *Code Generation; Deep Learning; Natural Language Processing, Ontology, SPARQL, Computational Intelligence, Artificial Intelligence*

1. INTRODUCTION

Research on deep learning-based code generation from natural language explores the application of advanced neural network architectures, particularly deep learning models, to automatically translate human-readable natural language specifications into executable code. This interdisciplinary approach leverages deep learning techniques such as transformers to enhance the understanding of complex language structures, enabling more accurate and context-aware code synthesis. The research aims to streamline software development processes, making them more efficient, accessible, and collaborative by minimizing the gap between high-level requirements and actual implementation through the power of deep learning algorithms [1]. Intellicode Compose is a code generation tool that employs transformer-based models to enhance the efficiency of code writing. Leveraging advanced transformer architectures, such as OpenAI's GPT, Intellicode Compose assists developers by suggesting and generating code snippets based on natural language queries and contextual understanding. This innovative approach accelerates coding tasks, providing developers with intelligent suggestions and improving overall productivity by harnessing the power of transformer models for context-aware and accurate code generation. [2] Deep code comment generation involves using advanced machine learning techniques, often deep neural networks, to automatically generate descriptive comments for source code. By analyzing code structure and functionality, these models produce human-readable comments that explain the purpose, logic, and usage of code snippets. This enhances code documentation, aiding developers in understanding, maintaining, and collaborating on projects. The goal is to automate the often time-consuming process of writing comments, improving code comprehensibility and fostering better communication within development teams. [3] The problem statement involves developing a computational intelligence model for code generation from natural language. This research aims to create an advanced system that interprets human-readable requirements and transforms them into executable code using computational intelligence techniques. The focus is on addressing challenges in accurate translation and improving the efficiency of generating code from diverse and complex natural language inputs. [4] A deep learning model for source code generation employs neural network architectures to automatically

produce program code. This approach involves training the model on large datasets of code examples, enabling it to learn syntax, semantics, and programming patterns. The model's deep layers' capture intricate relationships, facilitating accurate code synthesis based on given specifications. This innovative application of deep learning streamlines software development, automating portions of the coding process and demonstrating the potential for AI to contribute significantly to code generation tasks. [5] Utilizing code generation to enhance code retrieval and summarization. By intertwining the processes of generating and summarizing code, the model learns to improve both tasks simultaneously. This approach leverages synergies between code synthesis and summarization, fostering advancements in code-related information retrieval and concise summarization using a dual learning framework [6] This study investigates automated source code generation and auto-completion through deep learning. By comparing and discussing various language model-related approaches, the research aims to identify the most effective techniques for enhancing code completion and generation processes. The focus is on leveraging state-of-the-art language models to improve accuracy and efficiency in coding tasks [7] Automated encoding of clinical documents using natural language processing (NLP) involves the development of systems that extract and encode relevant information from medical texts. NLP algorithms analyze unstructured clinical narratives, identifying key entities, relationships, and concepts, and convert them into structured, machine-readable formats. This automated encoding streamlines healthcare data management, enabling efficient retrieval, analysis, and integration of patient information. The goal is to enhance clinical decision support systems, facilitate research, and improve overall healthcare outcomes by leveraging NLP to transform free-text clinical documentation into structured and actionable data [8] A programming language interface for describing transformations and code generation provides a structured means to specify how source code should be transformed or generated. This interface typically includes syntax and constructs that allow developers to articulate the desired changes in code. By offering a higher-level abstraction, it enhances the expressiveness and readability of transformation instructions, facilitating more intuitive and efficient code manipulation. Such interfaces are crucial for tools and frameworks that automate code transformations, enabling developers to articulate complex changes effectively. [9] A programming language interface for transformations and code generation provides a structured way to articulate changes in source code. This interface enhances expressiveness, allowing developers to specify desired code modifications effectively, facilitating automated transformation tools and frameworks for more efficient and readable code manipulation. [10] Automatic Multilingual Code Generation involves using NLP techniques to automatically generate code in multiple programming languages from human-readable natural language specifications. This innovative approach aims to make code generation accessible across different linguistic contexts, enhancing the versatility and global applicability of software development processes. [11] Transformer-based code generation techniques. By examining the output code produced by transformer models, the research aims to identify and analyze common code smells indicators of potential design issues or inefficiencies. Through a systematic examination of generated code, the study provides insights into the strengths and weaknesses of transformer-based approaches, helping developers and researchers understand the challenges and opportunities in improving the quality and maintainability of code produced by these advanced natural language processing models. [12] Seq2Code is a transformer-based encoder-decoder model designed for Python source code generation. Leveraging the transformer architecture, it interprets natural language descriptions and converts them into Python code. By capturing contextual dependencies, Seq2Code excels in understanding the intricacies of coding tasks, facilitating accurate and context-aware code synthesis. This model significantly streamlines the process of generating Python code from human-readable specifications, showcasing the potential of transformer-based approaches in advancing code generation tasks. [13]

2. MATERIALS AND METHOD

2.1. Alternative parameters: Seq2Seq, SNM, Tree2Tree, TRANX, Coarse-to-Fine
2.2. Evaluation parameters: Efficiency, Readability, Cost, Ease of Use
2.3. Efficiency: Efficiency refers to the ability to achieve maximum output with minimum resources, time, or effort. In various contexts, efficiency implies optimizing processes, minimizing waste, and enhancing productivity. It can be measured by the ratio of output to input and involves streamlining operations, reducing redundancy, and achieving desired results with the least amount of resources or time expenditure. Efficiency is a key concept in fields ranging from business and engineering to technology and environmental sustainability, where the goal is to attain optimal outcomes while conserving resources and minimizing any unnecessary or extraneous activities.
2.4. Readability: "Readability" refers to the ease with which written text can be understood by a reader. It is a measure of how easily a person can comprehend a piece of writing. Several factors contribute to readability, including the complexity of sentence structures, vocabulary level, and overall clarity of expression. There are various formulas and methods used to assess readability, such as the Flesch-Kincaid Grade Level, Gunning Fog

Index, and Coleman-Liau Index. These tools analyze aspects like sentence length and syllable count to estimate the educational level required to understand a given text. The goal of improving readability is to make written information more accessible and understandable to a wider audience, regardless of their education level or background. This is particularly important in fields like journalism, education, and technical writing, where effective communication is crucial.

2.5. Cost: Financial Cost: This is the most common understanding of cost, referring to the monetary expenditure associated with acquiring goods or services. It includes the actual purchase price as well as additional expenses like taxes, fees, and shipping. Opportunity Cost: This concept refers to the value of the next best alternative forgone when a decision is made. In other words, it's the cost of choosing one option over another. Time Cost: This represents the amount of time required to perform a task or produce a product. Time is often considered a valuable resource, and the time spent on one activity could have been used for something else.

2.6. Ease of Use: "Ease of use" refers to the degree of simplicity and user-friendliness of a product, system, or interface. It is a measure of how easily and efficiently a person can interact with and navigate through a particular item or system without encountering confusion, frustration, or the need for extensive training.

2.7. Seq2Seq (Sequence-to-Sequence): Seq2Seq models are commonly used in NLP for tasks such as machine translation, summarization, and code generation. In the context of code generation, a Seq2Seq model takes a sequence of natural language (e.g., a code description) as input and produces another sequence (e.g., the generated code).

2.8. SNM (Syntax-Guided Neural Machine Translation): SNM focuses on incorporating syntax information into neural machine translation models. In the context of code generation, it aims to leverage the syntax of programming languages to improve the quality and correctness of generated code.

Tree2Tree: This approach likely involves the use of tree structures to represent both the input (natural language) and output (code) during the generation process. Tree-based models can capture hierarchical relationships in the data, which is beneficial for representing the structure of code.

2.9. TRANX (Tree-Structured Neural Code Generator): TRANX is a specific model designed for translating natural language into code. It utilizes a tree-structured neural network to capture the hierarchical structure inherent in programming languages.

2.10. Coarse-to-Fine: Coarse-to-Fine approaches involve a multi-stage process where the model initially generates a coarse representation and then refines it to produce a more detailed and accurate output. This approach can be useful for complex tasks like code generation, where the output needs to adhere to both high-level and low-level syntax rules.

Method: COPRAS (Complex Proportionality Assessment) is one of the most used (MCTM) methods, and the ratio of the best solution Determining the solution with the best rate in the set of possible alternatives by Provides a better alternative Bad Solution. This technique has Decision making problems Various to solve used by researchers [14]. The COPRAS-G method requires identifying selection criteria; evaluating information related to these criteria, and developing methods to evaluate Meeting the participant's needs Criteria for doing in order to assess the overall performance of the surrogate. Decision analysis involves a Decision Maker (DM) Situation to do consider a particular set of alternatives and select one among several alternatives, usually with conflicting criteria. For this reason, the developed complexity proportionality assessment (COPRAS) method can be used [15]. In 1996 in Lithuania COPRAS (Complex Proportion evaluation) method was developed. Construction economics, real estate and management. One of the articles assesses the risks involved in construction projects. The assessment is based on various multi-objective assessment methods. The risk assessment indices are selected considering the interests, objectives and factors of the countries that influence the construction efficiency and real estate price increase [16] to describe and consider the task model. Complex Proportionality Assessment (COPRAS) Method Similar to any many other criteria will make the decision (MCDM) tool, first Proposed COBRAS method of several related criteria basically for alternatives Used to prioritize criterion weights. This method is better and Worst-Best Solutions Best decision considering Selecting alternatives [17]. Cobras approach is used for device tool choice; Because of this the triangle Ambiguous numbers are selected their computational performance. Three area specialists are selected to assign weights and by way of combining the fuzzy cobra's method, System 1 (MC1) and device 2(MC2) similarly are ranked, with way of ma chine three and four. -based totally approach is utilized in mixture with fuzzy. COPRAS assess the complexity of consumer dating management (CRM) performance. A combined choice matrix is obtained from a panel of 20 specialists offered 3 options with set, and 5 criteria Assessment are done [18]. COPRAS to resolve MCDM issues, wherein the weights of the criteria and Performance ratings of alternatives are absolute Based on linguistic terms are calculated. Comparison of criteria Importance calculated and Cobras method become used to assess renovation strategies [19]. This have a look at ambitions to develop the impact of latest overall performance metrics in TPM and COPRAS in an ambiguous context Primarily multi-criteria selection based on opinions Use the do method. Looseness of paper is prepared as follows. Section 1 provides an overview of the disruption and a literature review. Section 2 focuses on the Cobras-G approach and the corresponding literature review. Sections three and four introduce the core principles of the Cobras-G methodology, emphasizing its utilization based on the recommended

approach, which is described as COPRAS-G. This complex proportional estimation approach employs numerical data represented using the Grey Systems Theory framework. The Cobras-G concept approach is rooted in Grey Systems Theory programs, real-world decision-making scenarios, and duration-based standard values. Diploma [20]. COPRAS method changed into the most relevant social media platform Rank and choose is used. Proposed Applicability of the structure We proved and proved the character [21].COPRAS (Complex Proportionality Assessment) To examine Cumulative of an alternative Performance, it is essential become aware of the maximum vital criteria, examine the options and compare the facts Depending on those criteria to fulfill the wishes of the DMs to compare grades evaluation involves a situation in which a DM must pick amongst several downloaded alternatives given a selected set of commonly conflicting standards. For this motive, the developed complex proportionality evaluation (COPRAS) method can be used in real situations, alternatives The criteria for assessment are vague is related to the factor, And the values of the standards are real Cannot be expressed with numbers [22].

3. RESULTS AND DISCUSSION

TABLE 1. Natural language processing for code generation

Approaches	Efficiency	Readability	Cost	Ease of Use
Seq2Seq	0.75	0.90	500.00	8.50
SNM	0.85	0.80	600.00	7.20
Tree2Tree	0.70	0.85	450.00	9.00
TRANX	0.88	0.88	550.00	7.80
Coarse-to-Fine	0.82	0.75	480.00	8.20

Table 1 shows compare above values Efficiency: TRANX has the highest efficiency (0.88), indicating that it performs well in generating code. Tree2Tree has the lowest efficiency (0.70), suggesting it may not be as effective in this aspect. Readability: Seq2Seq and TRANX have high readability scores (0.90 and 0.88, respectively), suggesting that the generated code from these approaches is easy to understand. Coarse-to-Fine has the lowest readability score (0.75), indicating that its generated code might be less clear. Cost: Tree2Tree has the lowest cost (450.00), making it the most cost-effective option. SNM has the highest cost (600.00), indicating it might be relatively more expensive. Ease of Use: Tree2Tree has the highest ease of use score (9.00), suggesting that it is the most user-friendly approach. SNM has the lowest ease of use score (7.20), indicating that it may be less user-friendly.

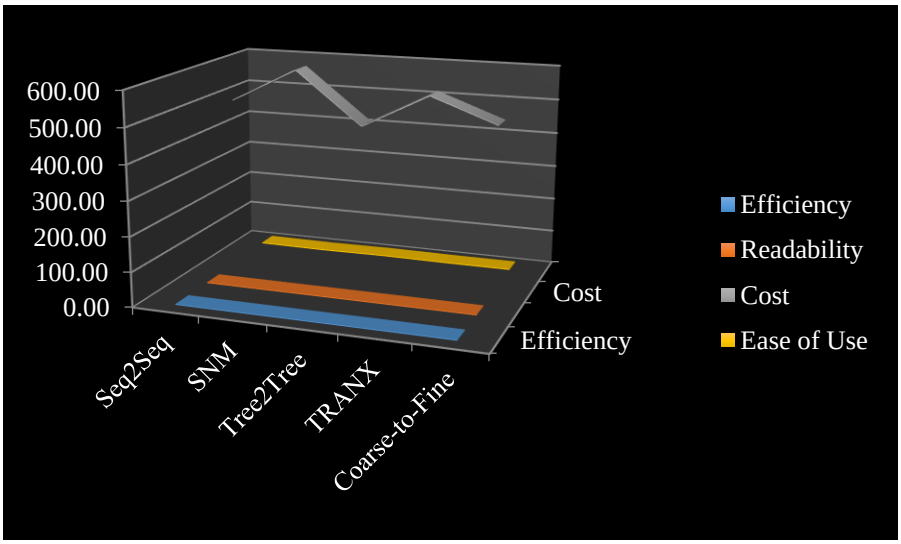


FIGURE 1. Natural language processing for code generation

Figure 1 illustrate the graphical representation of Natural language processing for code generation

TABLE 2. Normalized Data

Normalized Data			
Efficiency	**Readability**	**Cost**	**Ease of Use**
0.1875	0.2153	0.1938	0.2088
0.2125	0.1914	0.2326	0.1769
0.1750	0.2033	0.1744	0.2211
0.2200	0.2105	0.2132	0.1916
0.2050	0.1794	0.1860	0.2015

Table 2 shows the explanation of normalized data Efficiency: The normalized values suggest that Approach 4 has the highest efficiency, while Approach 3 has the lowest. Readability: Approach 1 has the highest normalized readability score, and Approach 5 has the lowest. Cost: Approach 3 has the lowest normalized cost, while Approach 2 has the highest. Ease of Use: Approach 3 has the highest normalized ease of use score, and Approach 2 has the lowest. By normalizing the data, it becomes easier to compare the relative performance of different approaches across the four criteria, as all values are on a consistent scale.

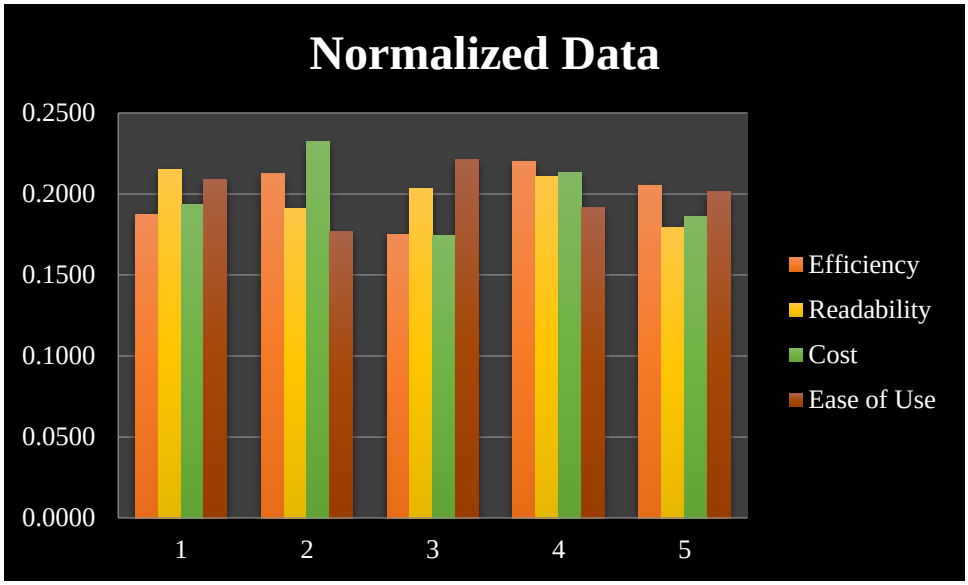


FIGURE 2. Normalized Data

Figure 2 illustrate the graphical representation of Normalized Data.

TABLE 3. Weightages

Weight			
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25

Table 3 shows weight ages the interpretation of these weights suggests that all five approaches (Seq2Seq, SNM, Tree2Tree, TRANX, Coarse-to-Fine) are considered equally important in the assessment of NLP for code generation.

TABLE 4. Weighted normalized decision matrix

Weighted normalized decision matrix			
0.05	0.05	0.05	0.05
0.05	0.05	0.06	0.04
0.04	0.05	0.04	0.06
0.06	0.05	0.05	0.05
0.05	0.04	0.05	0.05

Table 4 shows weighted normalized decision matrix. The weights and normalized scores in the matrix are used to calculate a weighted sum or weighted average for each alternative. This allows decision-makers to compare and rank the alternatives based on their overall performance, considering the relative importance of the criteria. To calculate the overall performance score for the first alternative, would multiply each criterion's score by its respective weight, and then sum the products: Overall Performance (Alternative1) = (0.02* Weight1) +(0.06* Weight2) +(0.06* Weight3) +(0.05* Weight4)

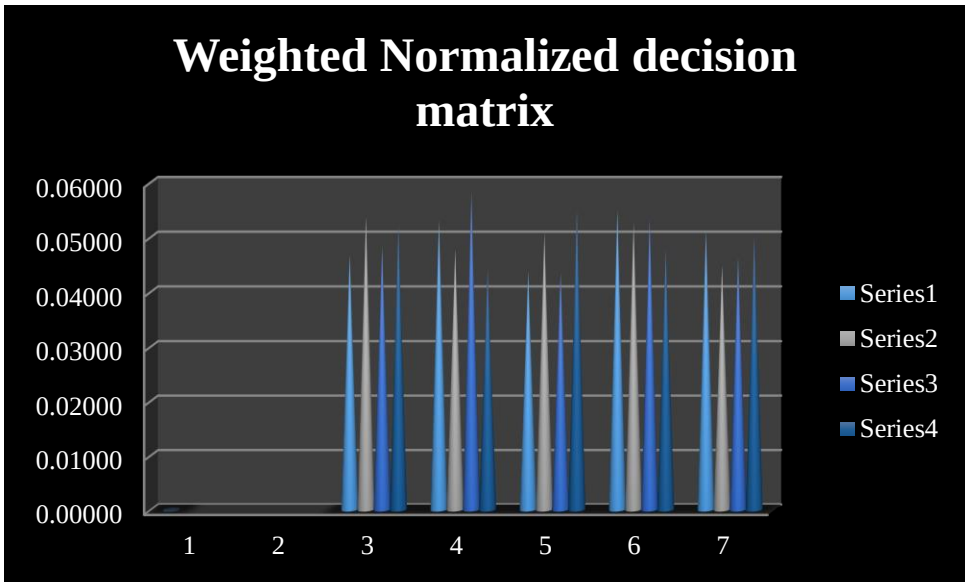


FIGURE 3. weighted normalized decision matrix

Figure 3 illustrate the graphical representation of weighted normalized decision matrix

TABLE 5. Natural language processing for code generation Bi, Ci, Min (Ci)/Ci

Bi	**Ci**	**Min(Ci)/Ci**
0.101	0.101	0.9624
0.101	0.102	0.9464
0.095	0.099	0.9797
0.108	0.101	0.9573
0.096	0.097	1.0000

Table 5 shows Bi, Ci, Min (Ci/Ci) explain these values Bi: This column represents a variable denoted by "Bi." Each row has a specific value for Bi. Ci: This column represents another variable denoted by "Ci." Each row has a specific value for Ci. Min (Ci)/Ci: This column shows the ratio of the minimum value of Ci to Ci in each row. The formula for each row is: Min(Ci)/Ci let's interpret the values in the table: The minimum value of Ci in this row is the same as Ci itself (0.101), and the ratio is 0. 9624.The minimum value of Ci is 0.099, and the ratio is 0.9797. The minimum value of Ci is 0.099, and the ratio is 0.9797. The minimum value of Ci is 0.101, and the ratio is 0.9573. The minimum value of Ci is 0.097, and the ratio is 1.0000, indicating that the minimum value equals Ci. These values provide insights into relationships between Bi and Ci, with the focus on the impact of the minimum Ci value on the ratio. The specific significance of these values depends on the context and purpose of the analysis.

TABLE 6. Final Result of Natural language processing for code generation

Approaches	Qi	Ui	Rank
Seq2Seq	0.200	96.9022	2
SNM	0.199	96.2314	4
Tree2Tree	0.196	94.8024	5
TRANX	0.206	100.0000	1
Coarse-to-Fine	0.199	96.5530	3

Table 6 shows final result of NLP for code generation Qi: This column represents a numerical score or measure labeled as "Qi" for each approach. The specific meaning or calculation of "Qi" is not provided in the table, but it seems to be a quantitative measure of some aspect of performance or quality associated with each approach. Ui: This column represents another numerical score or measure labeled as "Ui" for each approach. Like "Qi," the specific meaning or calculation of "Ui" is not provided, but it appears to be a quantitative metric associated with the performance of each approach.

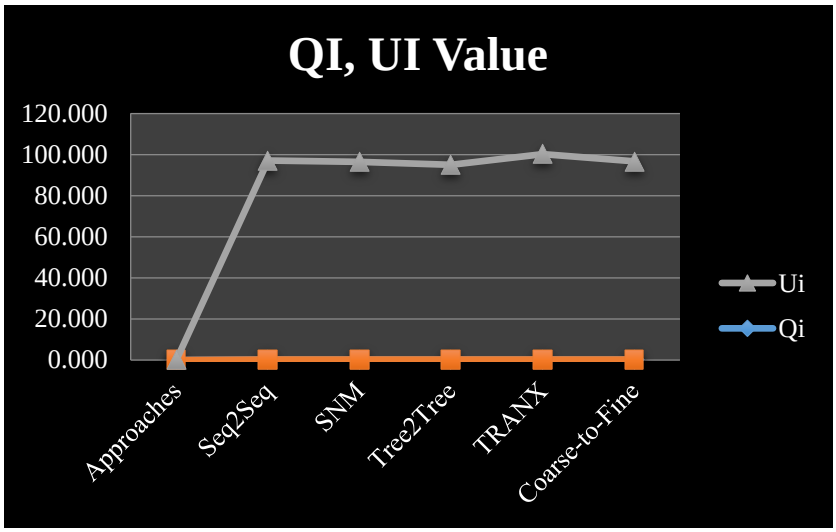


FIGURE 4. NLP for code generation Qi, Ui value

Figure 4 illustrate the graphical representation of Qi, Ui value.

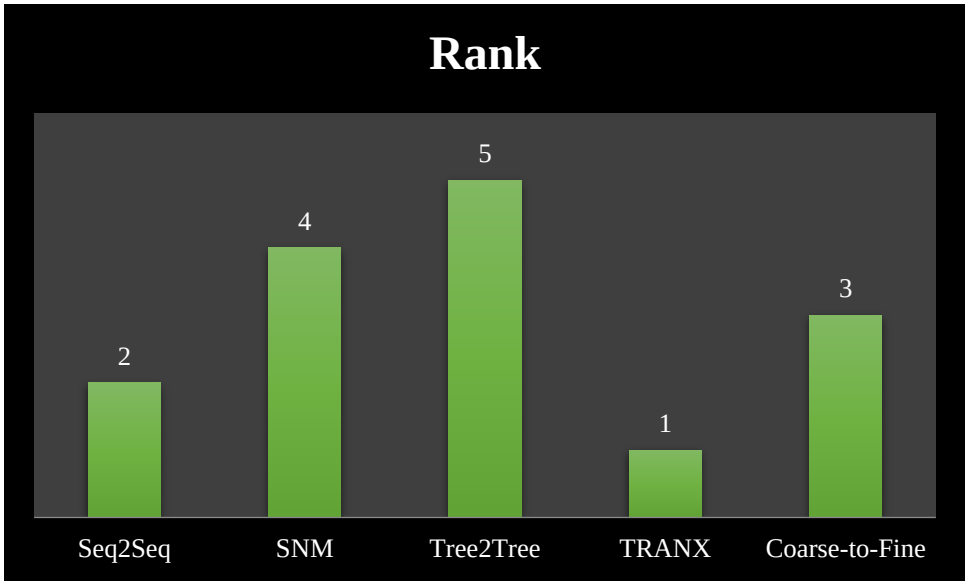


FIGURE 5. Rank

Figure 5 Shows ranking of NLP for code generation, TRANX is got the first rank whereas is the Tree2 tree is having the lowest rank.

4. CONCLUSION

The integration of Natural Language Processing (NLP) into code generation holds tremendous promise for revolutionizing the landscape of software development. The ability to bridge the gap between human language and programming languages not only enhances developer productivity by allowing for more intuitive expression of intentions but also opens up programming to a broader audience. The prospect of reduced barriers for non-programmers, who can articulate their ideas in natural language and have them translated into executable code, heralds a new era of inclusivity in software development. While challenges such as ambiguity and optimization persist, ongoing research and development suggest that NLP-based code generation tools will become indispensable in modern Integrated Development Environments (IDEs), offering invaluable assistance, streamlining workflows, and contributing to the evolution of a more accessible and efficient programming ecosystem. Ethical considerations must guide this development to ensure fairness and transparency, but the overall trajectory points toward a future where NLP seamlessly integrates into the daily practices of developers, shaping the way we conceive, write, and collaborate on code

REFERENCES

[1]. Zhu, Jiaqi, and Mingzhu Shen. "Research on Deep Learning Based Code generation from natural language Description." In *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pp. 188-193. IEEE, 2020.

[2]. Svyatkovskiy, Alexey, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. "Intellicode compose: Code generation using transformer." In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1433-1443. 2020.

[3]. Hu, Xing, Ge Li, Xin Xia, David Lo, and Zhi Jin. "Deep code comment generation." In *Proceedings of the 26th conference on program comprehension*, pp. 200-210. 2018.

[4]. Kulkarni, A. B., S. S. Karandikar, P. A. Bamhore, S. R. Gawade, and D. V. Medhane. "Computational Intelligence Model for Code Generation from Natural Language Problem Statement." In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1-6. IEEE, 2018.

[5]. Tiwang, Raymond, Timothy Oladunni, and Weifeng Xu. "A deep learning model for source code generation." In *2019 SoutheastCon*, pp. 1-7. IEEE, 2019.

[6]. Ye,Wei, Rui Xie, Jinglei Zhang, Tianxiang Hu, Xiaoyin Wang, and Shikun Zhang. "Leveraging code generation to improve code retrieval and summarization via dual learning." In *Proceedings of the Web Conference 2020*, pp. 2309-2319. 2020.

[7]. Cruz-Benito, Juan, Sanjay Vishwakarma, Francisco Martin-Fernandez, and Ismael Faro. "Automated source code generation and auto-completion using deep learning: Comparing and discussing current language model-related approaches." *AI* 2, no. 1 (2021): 1-16.

[8]. Friedman, Carol, Lyudmila Shagina, Yves Lussier, and George Hripcsak. "Automated encoding of clinical documents based on natural language processing." *Journal of the American Medical Informatics Association* 11, no. 5 (2004): 392-402.

[9]. Rudy, Gabe, Malik Murtaza Khan, Mary Hall, Chun Chen, and Jacqueline Chame. "A programming language interface to describe transformations and code generation." In *Languages and Compilers for Parallel Computing: 23rd International Workshop, LCPC 2010, Houston, TX, USA, October 7-9, 2010. Revised Selected Papers 23*, pp. 136-150. Springer Berlin Heidelberg, 2011.

[10]. Bajwa, Imran Sarwar, M. Shahid Naveed, and M. Abbas Choudhary. "Natural Language Processing based Automatic Multilingual Code Generation."

[11]. Siddiq, Mohammed Latif, Shafayat H. Majumder, Maisha R. Mim, Sourov Jajodia, and Joanna CS Santos. "An Empirical Study of Code Smells in Transformer-based Code Generation Techniques." In *2022 IEEE 22nd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pp. 71-82. IEEE, 2022.

[12]. Laskari, Naveen Kumar, K. Adi Narayana Reddy, and M. Indrasena Reddy. "Seq2Code: Transformer-Based Encoder-Decoder Model for Python Source Code Generation." In *Congress on Intelligent Systems*, pp. 301-309. Singapore: Springer Nature Singapore, 2022.

[13]. Laskari, Naveen Kumar, K. Adi Narayana Reddy, and M. Indrasena Reddy. "Seq2Code: Transformer-Based Encoder-Decoder Model for Python Source Code Generation." In *Congress on Intelligent Systems*, pp. 301-309. Singapore: Springer Nature Singapore, 2022.

[14]. Yazdani, Morteza, Ali Alidoosti, and EdmundasKazimierasZavadskas. "Risk analysis of critical infrastructures using fuzzy COPRAS." Economic research-Ekonomskaistraživanja 24, no. 4 (2011): 27-40.https://doi.org/10.1080/1331677X.2011.11517478

[15]. Aghdaie, Mohammad Hasan, Sarfaraz HashemkhaniZolfani, and EdmundasKazimierasZavadskas. "Market segment evaluation and selection based on application of fuzzy AHP and COPRAS-G methods." Journal of Business Economics and Management 14, no. 1 (2013): 213-233.https://doi.org/10.3846/16111699.2012.721392

[16]. Kildienė, Simona, Arturas Kaklauskas, and EdmundasKazimierasZavadskas. "COPRAS based comparative analysis of the European country management capabilities within the construction sector in the time of crisis." Journal of Business Economics and Management 12, no. 2 (2011): 417-434.

[17]. Das, Manik Chandra, Bijan Sarkar, and Siddhartha Ray. "A framework to measure relative performance of Indian technical institutions using integrated fuzzy AHP and COPRAS methodology." Socio-Economic Planning Sciences 46, no. 3 (2012): 230-241.https://doi.org/10.1016/j.seps.2011.12.001

[18]. Dhiman, Harsh S., and Dipankar Deb. "Fuzzy TOPSIS and fuzzy COPRAS based multi-criteria decision making for hybrid wind farms." Energy 202 (2020): 117755.https://doi.org/10.1016/j.energy.2020.117755

[19]. Fouladgar, Mohammad Majid, Abdolreza Yazdani-Chamzini, Ali Lashgari, EdmundasKazimierasZavadskas, and ZenonasTurskis. "Maintenance strategy selection using AHP and COPRAS under fuzzy environment." International journal of strategic property management 16, no. 1 (2012): 85-104.https://doi.org/10.3846/1648715X.2012.666657

[20]. TuranogluBekar, Ebru, Mehmet Cakmakci, and Cengiz Kahraman. "Fuzzy COPRAS method for performance measurement in total productive maintenance: a comparative analysis." Journal of Business Economics and Management 17, no. 5 (2016): 663-684.https://doi.org/10.3846/16111699.2016.1202314

[21]. Zolfani, Sarfaraz Hashemkhani, Nahid Rezaeiniya, Mohammad Hasan Aghdaie, and EdmundasKazimierasZavadskas. "Quality control manager selection based on AHP-COPRAS-G methods: a case in Iran." Economic research-Ekonomskaistraživanja 25, no. 1 (2012): 72-86.https://doi.org/10.1080/1331677X.2012.11517495

[22]. Tavana, Madjid, Ehsan Momeni, Nahid Rezaeiniya, Seyed Mostafa Mirhedayatian, and Hamidreza Rezaeiniya. "A novel hybrid social media platform selection model using fuzzy ANP and COPRAS-G." Expert Systems with Applications 40, no. 14 (2013): 5694-5702.https://doi.org/10.1016/j.eswa.2013.05.015

[23]. Kouchaksaraei, RamtinHaghnazar, Sarfaraz HashemkhaniZolfani, and Mahmood Golabchi. "Glasshouse locating based on SWARA-COPRAS approach." International Journal of Strategic Property Management 19, no. 2 (2015): 111-122.https://doi.org/10.3846/1648715X.2015.1004565