International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Smart Tutor: An AI tutor using Natural Language Processing

S. Tribhuvan[1], Amritanshu Kaundilya[2], Siddhesh Sasane[3], Ujjwal Choudhary[4], Saudagar Sushil[5]

[1]Professor, Dept. of Cloud Computing and Big Data, P.Dr.V.V.P. Institute of Technology and Engineering, Loni, Maharashtra, India

[2,3,4,5]Final year Diploma Student, P.Dr.V.V.P. Institute of Technology and Engineering, Loni, Maharashtra, India

---

Abstract - Utilizing AI and NLP, the Smart Tutor system offers students individualized learning support. It is a state-of-the-art teaching platform. Through sophisticated processing of student queries and the delivery of precise and contextual responses, the system is aimed to mimic human-like teaching. The knowledge-driven tutoring framework it builds includes features including answer extraction, model training, document feeding, and preprocessing. The Smart Tutor improves education by facilitating efficient question answering, adaptable pedagogy, and real-time interaction. Offering scalable, accessible, and efficient education support, this system aspires to bridge the gap between modern intelligent tutoring systems and traditional classroom learning.

Keywords: Artificial Intelligence (AI), Natural language Processing (NLP), Machine Learning, Speech Recognition, Adaptive Learning

1.INTRODUCTION

Culture, identity, and international communication are all profoundly impacted by programming languages, which are at the very heart of human communication. To thrive in today's globally integrated economy, further one's education, and climb the corporate ladder, one must acquire fluency in multiple programming languages. Traditional approaches to teaching programming languages, on the other hand, tend to use inflexible lesson plans, boring drills, and a one-size-fits-all approach that might not work for some students. A new age of innovation has dawned in programming language learning with the arrival of digital technology and Artificial Intelligence (AI). Intelligent, interactive, and adaptive systems are revolutionizing the way humans acquire linguistic skills.

Education is only one of several fields that has been profoundly affected by the advent of AI. A number of its subfields, including ML, Deep Learning, Speech Recognition, and Natural Language Processing (NLP), have made it possible for computers to comprehend, interpret, and produce meaningful programming language. A virtual tutor that can analyze voice, correct pronunciation, assess grammar, and provide real-time feedback to learners is created using these technologies as part of the concept of learning a programming language through an AI system. This AI-based solution allows users to learn whenever and wherever they want through interactive interactions with sophisticated algorithms, unlike traditional learning tools.

An intelligent and data-driven method that customizes the experience for each learner is offered by integrating AI into programming language learning. The system finds out where the student is doing well and where they need improvement by collecting and analyzing data like their vocabulary usage, spelling errors, and attempts at pronouncing words. Afterwards, it dynamically adjusts the curriculum to display individual-specific lessons, exercises, and levels of difficulty based on their development. If a student is having trouble with verb conjugation or pronouncing certain sounds, for instance, the AI system can mimic a human instructor by creating targeted practice sessions and providing immediate feedback on how to improve. Learning the target programming language becomes easier and faster with each round of this feedback and adaptation loop.

When the user inputs text or speech, the system processes and understands it using sophisticated natural language processing techniques. In order to mimic human communication, it understands language context, finds mistakes, and reacts correctly. Students may practice their speaking and listening abilities in a safe, non-judgmental setting with Chabot-based engagement by participating in realistic conversations. The system can analyze the fluency and correctness of spoken input with voice recognition technology, and text-to-speech modules can improve listening comprehension and pronunciation modeling by producing native-like audio outputs. When put together, these tools create an immersive learning environment that improves proficiency in reading, writing, hearing, and speaking the programming language.

There is a huge and ever-changing future for AI-assisted programming language learning. Future systems may employ visual clues, emotion recognition, and cultural knowledge to further enhance communication's naturalness and human-likeness, thanks to developments in multimodal AI and deep learning models. Think of a system that helps students become more fluent in not just words and grammar but also in context, via the interpretation of nonverbal cues like tone of voice and facial expressions. Programming language education will be reshaped by these developments, which will have an even greater impact on worldwide collaboration, cross-cultural communication, and international business.

Last but not least, the advancements in education and communication technologies brought about by the Learning Programming language through AI System are nothing short of revolutionary. This system offers a versatile, personalized, and engaging learning experience by merging the analytical capabilities of AI with the expressiveness and creativity of human programming language. Intelligent data analysis guarantees continual progress, real-time feedback boosts engagement, and learners are empowered to overcome traditional obstacles. In the end, programming language learning systems powered by AI not only streamline the process of learning new languages, but they also create a worldwide community where everyone can communicate, comprehend, and learn.

[1] Eduard Frankford et al. introduced the idea of using ChatGPT as an AI tutor in the Artemis APAS and found out all about the possible benefits and drawbacks of this use. There were clear limitations to the AI-Tutor, despite its benefits like scalability and quick feedback. Among these were issues such as technical limitations, students' possible over-reliance, the lack of a personal touch, operational vulnerabilities connected to API317ICSE-SEET '24, April 14–20, 2024, Lisbon, Portugal Eduard Frankford, Clemens Sauerwein, Patrick Bassner, Stephan Krusche, and Ruth Breu, and a lack of interactive dialog.Careful installation and continual refining are necessary for AI-Tutors to fully realize their immense potential in programming instruction. The importance of further research in this area is shown by this exploration, which highlights the necessity to find a balance between technology progress and the irreplaceable human element of education.Improving the accuracy of AI-Tutor's feedback, making it more interactive, refining the user interface, and fixing the problems with the token limit and prompt engineering should be the priorities of future efforts.Investigating more robust models, such as GPT-4, has the ability to enhance the feedback quality even further. The results of this study lay the groundwork for future investigations into the novel area where artificial intelligence and education converge.

[2] Woo-Hyun Kim et al. created a new kind of personalized AI tutor that can really boost a student's grades. We built the AI instructor as a system that extends the Deep ART network and integrates three DLNs to take the learner's current condition and preferences into account. A commercialized mobile app that teaches the Korean programming language was used to train the suggested AI instructor with 800,000 training sets. The results of our experiments demonstrate that the suggested AI tutor can recommend appropriate learning materials to students based on what is taught in the conventional school curriculum. Besides this, we showed that our AI tutor can recommend appropriate learning contents even for learners to whom the standard curriculum cannot make suggestions. In order to gauge the efficacy of a learning program over time, we intend to gather data on each student's progress based on their interactions with the suggested AI tutor via the mobile app platform.

[3] Paulius Danenas et al. studied model-to-model transformations using natural programming language processing in 2022. Article discusses research on new short text processing methods. This study was inspired by our prior work on model-to-model transformations[6,7], but process mining, aspect-based sentiment analysis, and conversational interfaces that process brief messages like commands may also benefit from this type of text processing. To function properly, natural language processing (NLP) must address commonalities across all of these areas, such as the necessity to recognize verbs and nouns and the lack of context for automated processing. This study discusses problems extracting relation tuples from process and system needs models with activity-like expressions. As indicated in Section III, applied modeling approaches and many ambiguities are not addressed in typical NLP processing toolkits, making the problem difficult to handle. Poor modeling practices include processing conjunction or disjunctive clauses and abbreviations and acronyms. We studied state-of-the-art implementations through our study and integrated them using our formal grammar-based extraction to produce prototypes to solve the difficulties highlighted in this work. We designed and tested our own bidirectional LSTM-CRF tagging tools using BERT and ELMO embeddings at the input layer and input corpora augmentations.

The second section of this research analyzes prior studies classified as Litereature Surveys. Section 3 provides a detailed description of the proposed methodology. Part 4 delves into the experimental examination, while Section 5 considers alternative revisions before concluding the essay with a statement on the existing plan.

2. LITERATURE SURVEY

[4] Shengyi Jiang et al. introduce PTMs for the Khmer Programming Language for the first time using BERT and ELECTRA. Considering the challenges presented by limited resources and the difficulty of compiling labeled data, we only apply the models to two downstream tasks, the dataset for one of which is self-constructed. The experimental results confirm the efficacy of our Khmer PTMs. We also investigate if completing word segmentation has a positive effect on subsequent tasks. Although the present Khmer word segmentation technique may provide some benefits, the gains are not considerable. We think that by making our models and datasets available to the community, we will progress Khmer natural language processing research.

[5] Paula Maddigan et al. developed The capacity to construct visualisations using natural programming languages has long been an aim in the field of data visualization. The creation of Natural Programming Language Interfaces has opened the path for progress in this area, making data visualisation more accessible to a wider range of users by allowing them to express their queries and analytical objectives in natural programming language. However, the process of accurately and reliably translating natural programming language inputs into visualisations (NL2VIS) has proven difficult to solve due to the difficulties of understanding natural programming language. This work presented a revolutionary end-to-end approach for translating free-form natural programming languages into visualisations utilizing cutting-edge Large Programming Language Models (LLMs).

[6] Nilanjana Raychawdhary et al. introduced This study highlights the transformative significance of transformer-based models in addressing sentiment analysis difficulties for low-resource African programming languages. Using the AfriSentiSemEval 2023 datasets, we fine-tuned sophisticated models such as AfriBERTa, mDeBERTAV3 base, and XLM-R, demonstrating their ability to achieve outstanding accuracy, weighted F1, precision, and recall in multilingual sentiment classification tasks. The findings emphasize the vital need of including varied linguistic elements during pretraining and fine-tuning in order to properly address the intricacies of African programming languages. This study demonstrates the feasibility of fine-tuning transformer-based models to overcome issues in multilingual sentiment analysis for African programming languages. Among the models tested, AfroXLMR displayed higher adaptability, establishing itself as the most effective model for sentiment analysis in resource-constrained

environments. In the future, we intend to expand the dataset by include more annotated African programming languages and dialects to improve sentiment analysis for a wider range of programming languages. We will also investigate simple training methods, such as knowledge distillation, to improve model performance in resource-constrained environments, as well as model optimization approaches, such as quantization, knowledge distillation, and federated learning, to improve the deployment efficiency of sentiment analysis models in computationally limited environments.

[7] Abdessamad Benlahbib et al. This study proposes a detailed comparative analysis of multiple methods used on the CoLA dataset for linguistic acceptability categorization, exhibiting a clear trend: as we went from classic approaches to more modern ones, the results dramatically improved. Initial approaches like POS tagging, CountVectorizer, and TF-IDF were ineffective, and even recent embeddings like FastText and ELMo produced mediocre results. However, the major performance boost came from the employment of transformer-based models, specifically DeBERTa V3, which produced outstanding results and validated its selection for ensembling trials. 10https://platform.openai.com/docs/models 258, Volume 6, 2025.Although data augmentation with ungrammatical sentences generated by Claude LLM did not result in further gains, it emphasized the ongoing difficulty of dealing with imbalanced datasets. Furthermore, in-context learning studies employing ChatGPT with 10- and 20-shot configurations yielded promising results but did not outperform transformer-based models.

[8] Marwan Omar et al. Narrate a survey of NLP robustness research in a consistent and systematic manner. We discovered several gaps in the literature and made recommendations for future research topics based on the various pieces of the NLP pipeline. As numerous real-world NLP projects have failed after deployment due to a lack of robustness, it is critical to investigate robustness as a multidimensional term that necessitates the creation of novel methodologies. We emphasize that newly developed approaches must solve spurious correlation difficulties and achieve high out-of-distribution accuracy in order to offer significant sensitivity to perturbations and, eventually, high precision in actual text classification scenarios. Overall, we believe that our research will serve as a new guide for the research community in terms of technique, metric, and dataset to utilize, as well as inspire further interest and work in this area to solve the many gaps.

[9] Csaba Veres et al. examined the challenge that deep learning neural models pose to traditional generative phrase structure theories of natural programming language, and demonstrated that the challenge was invalid because the arguments could be applied equally and absurdly to phrase structure in code. We suggest that our argument is more powerful and lasting than previous attempts to demonstrate that neural networks cannot do specific tasks. This sort of reasoning has an illustrious history in the discipline, where the publication of Minsky and Pap resulted in a significant reduction in research effort for decades. Veres: Large Programming Language Models are Not Models of Natural Programming Languages: They are Corpus Models pert's Perseptrons, which demonstrated fundamental computational constraints of current brain models [59]. More advanced models were eventually developed to address these restrictions, and the previously scathing criticisms were eliminated. Our argument is more compelling because it is based on the success of neural models. We contend that achieving high performance on arbitrary NLP tasks is irrelevant to theories of natural programming languages in general, and generative grammar in particular, because the same arguments apply equally to programming languages that are clearly the product of a generative grammar. As a result, we suggested that the term "programming language model" is deceptive.

[10] Luis Jose Gonzalez-Gomez et al. Proposed a comprehensive analysis of the present NLP landscape for skill extraction, prediction, and taxonomy development. The findings highlight NLP's versatility and depth across multiple areas, including clinical storytelling, talent management, and education. The authors demonstrated a paradigm leap from basic text analytics to comprehensive semantic understanding, which bridges the gap between raw data and structured taxonomies. NLP's interdisciplinary character, broad scope, and increasing capabilities demonstrate the field's ability to affect the future of knowledge management and information extraction. Throughout this inquiry, NLP's potential for creating dynamic skill taxonomies becomes clear. However, the true innovation comes in the possibilities for further refining and expanding these methodologies, notably in projecting skill evolution and relevance in a continuously changing job market. The ability of NLP to foresee emerging or niche abilities from large textual datasets suggests that it has the potential to shape educational initiatives, advise hiring practices, and guide professional development pathways in the future.

[11] Jinzhao Zhou et al. Narrate BELT is made up of a novel D-Conformer architecture for encoding EEG into discrete representations and a bootstrapping training mechanism for learning EEG representations aligned with programming languages. Our results suggest that using natural programming language supervision to promote semantic EEG representation learning is an effective method. This is backed by significant improvements in a variety of EEG decoding tasks, including EEG-to-word classification, EEG-to-sentence decoding, and sentiment analysis. The proposed method also stimulates further in-depth investigation and discussion of the critical problem of decoding thoughts into text, which could lead to a plethora of novel BCI applications. Despite the advances made, there is still space for further growth in translation precision and fluency without the implicit use of teacher-forcing evaluation. In the future, we intend to collect additional EEG data relevant to programming languages in order to train a broader EEG encoder and address the underlying issue of data scarcity in this study area.

[12] Ying Ma et al. Present This study reveals that AI-powered solutions, particularly those based on deep learning (RNNs) and natural language processing (NLP), can considerably improve individualized programming language learning experiences. The findings demonstrate that, by responding to individual student demands, AI models improve programming language proficiency more effectively than traditional, non-adaptive methods. Personalized lessons based on learner performance result in increased engagement, higher

grammatical accuracy, and improved speech clarity, all of which help to accelerate programming language acquisition.

[13] Tingting Huang et al. Describe an innovative approach to LCI that employs NLP-based prioritizing and granularity to guide LCT map development. The prioritization mechanism emphasizes specific landscape properties in the resulting LCT maps, while the granularity control changes the output from fine-grained to coarse-grained representations. The findings show that, while modifying granularity largely effects visual presentation, prioritization effectively highlights specific landscape qualities, providing flexible and efficient tools for governmental landscape planning and decision making.

3.METHODOLOGY

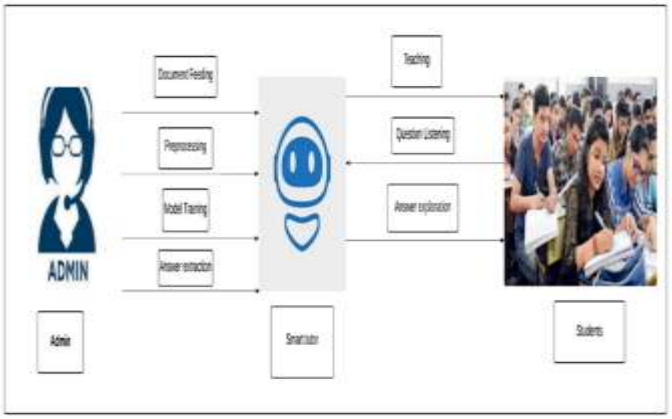

Fig 1: Overview Diagram

The proposed Smart Tutor system is designed as an interactive AI-based learning platform that combines Natural Language Processing (NLP), speech technologies, and information retrieval techniques to simulate a real teaching environment. When the system starts, the user (student) is presented with two options: (1) Have a Class or (2) Give a Test. Based on the selected option, the system proceeds into two distinct operational phases.

Phase 1: Interactive Teaching

In this phase, the system acts as a virtual AI tutor that delivers lecture content interactively. Once the student selects the *"Have a Class"* option, a robotic animated interface (GIF) is launched through the Eclipse environment, where lip synchronization is controlled using multithreading (speech ON/OFF threads). This creates a realistic teaching experience. A DOC file containing the study material is fed into the system. The text is preprocessed and converted into speech using text-to-speech libraries such as gTTS (Google Text-to-Speech), pyglite, and MukajEn, which enable real-time voice generation. The tutor reads the content paragraph by paragraph.

At the end of each paragraph, the system initiates an interactive feedback loop by asking the student whether they have understood the concept. A parallel speech recognition thread continuously listens to the student's response. If the student responds with "understood," the system proceeds to the next paragraph. Otherwise, if the student requests clarification "explain again", the same paragraph is repeated. This ensures adaptive and personalized learning.

Additionally, if the student asks a question during the session, the system retrieves the most relevant answer from the document using NLP-based similarity techniques. This is achieved through TF-IDF vectorization and cosine similarity, which measure the relevance between the student's query and document sentences.

The Smart Tutor system utilizes Natural Language Processing libraries such as TF-IDF Vectorizer and Cosine Similarity to extract relevant answers and evaluate student responses.

1. TF-IDF (Term Frequency – Inverse Document Frequency):

TF-IDF is used to determine the importance of a word in a document relative to a collection of documents. It assigns higher weight to words that are frequent in a document but rare across all documents.

The Term Frequency (TF), which represents how often a term appears in a document, is calculated using (Eq1):

$$TF(t,d) = \frac{f(t,d)}{\sum_k f(k,d)} \dots\dots\dots\dots\dots\dots\dots (Eq\ 1)$$

Where:

- $f(t,d)$= frequency of term t in document d
- $\sum_k f(k,d)$= total number of terms in document d

Thehe Inverse Document Frequency (IDF), which measures how unique or rare a term is across all documents, is calculated using (Eq 2)

$$IDF(t) = \log\left(\frac{N}{df(t)}\right) \dots\dots\dots\dots\dots\dots (Eq\ 2)$$

Where:

- N= total number of documents
- $df(t)$= number of documents containing term t

The final TF-IDF score is obtained by combining TF and IDF as shown in (Eq 3)

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t) \dots\dots\dots\dots\dots . (\text{Eq } 3)$$

2. Cosine Similarity:

Cosine similarity measures the similarity between two text vectors (such as a student query and a document sentence). It calculates the cosine of the angle between two vectors in a multi-dimensional space.

The cosine similarity is calculated using (Eq 4)

$$\text{Cosine Similarity} = \frac{A \cdot B}{||\ A\ ||\times||\ B\ ||} \dots\dots\dots\dots\dots\dots\dots\dots (Eq\ 4)$$

Where:

- $A \cdot B$= dot product of vectors A and B
- $||\ A\ ||, ||\ B\ ||$= magnitudes (lengths) of vectors A and B

The value ranges from **0 to 1**, where:

- **1** indicates maximum similarity
- **0** indicates no similarity

The sentence with the highest cosine similarity score is selected as the most relevant answer.

Phase 2: Assessment and Evaluation

In this phase, the system evaluates the student's understanding through an automated test mechanism. When the student selects the *"Give a Test"* option, a set of

predefined questions is retrieved from a structured CSV file. These questions are presented one by one to the student.
The system waits for the student's response, which is given via speech and terminated using a keyword such as "that's it". The spoken input is converted into text using speech recognition.

To evaluate the answer, the system compares the student's response with the model answer using the same NLP techniques applied in Phase 1, namely TF-IDF vectorization and cosine similarity. The similarity score determines how closely the student's answer matches the expected answer.

Each question is assigned a score based on this similarity value. All individual scores are then aggregated to compute the final score, which is displayed to the student at the end of the test. This provides an objective and automated evaluation system.

4. RESULTS AND DISCUSSION

The proposed Smart Tutor system is implemented using Python and NLP libraries to perform efficient answer extraction and evaluation. The system is developed and tested on a machine with an Intel Core i5 processor, 8 GB RAM, and standard storage capacity. For processing and storing learning content, lightweight data handling techniques are used, while TF-IDF vectorization and cosine similarity are applied for answer extraction.

Scalability Analysis of Answer Extraction

To evaluate the performance of the system, an analysis of answer extraction time is conducted based on the number of user queries processed. The system uses TF-IDF vectorization and cosine similarity to retrieve relevant answers from the given document.

Table 1 presents the relationship between the number of queries processed and the time required for answer extraction.

S. No	No. of Queries Processed	Answer Extraction Time (seconds)
1	10	0.42
2	20	0.88
3	30	1.35
4	40	1.72
5	60	2.05

Table 1: Answer Extraction Time Estimation

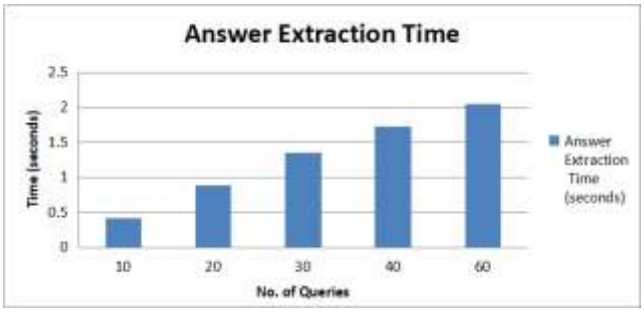

Fig 2: Analysis of Answer Extraction Time

Figure 2 illustrates the graphical representation of the same data. It clearly shows that as the number of queries increases; the answer extraction time also increases gradually. However, the growth is controlled and efficient, demonstrating the scalability of the proposed system.

The results indicate that the system is capable of handling a large number of queries with minimal delay. The slight increase in response time is due to the computation involved in vectorization and similarity calculation. Overall, the system ensures fast and accurate answer retrieval, validating the effectiveness of NLP techniques in real-time tutoring applications.

5. CONCLUSIONS

The Smart Tutor system represents a significant advancement in the field of intelligent education by integrating Artificial Intelligence and Natural Language Processing to deliver personalized and interactive learning experiences. It effectively reduces the limitations of traditional teaching methods by providing instant query resolution, detailed explanations, and continuous learning support. The system not only enhances student understanding and engagement but also supports educators by automating repetitive tasks. Overall, it serves as a scalable and efficient solution for modern education, making learning more accessible, flexible, and effective.

The system can be enhanced by adding voice-based interaction and multilingual support for better accessibility. It can also incorporate adaptive learning and advanced AI models to improve accuracy and personalization.

REFERENCES

[1] E. Frankford, C. Sauerwein, P. Bassner, S. Krusche and R. Breu, "AI-Tutoring in Software Engineering Education: Experiences with Large Programming language Models in Programming Assessments," 2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), Lisbon, Portugal, 2024, pp. 309-319, doi: 10.1145/3639474.3640061.

[2] W. -H. Kim and J. -H. Kim, "Individualized AI Tutor Based on Developmental Learning Networks," in IEEE Access, vol. 8, pp. 27927-27937, 2020, doi: 10.1109/ACCESS.2020.2972167.

[3] P. Danenas and T. Skersys, "Exploring Natural Programming language Processing in Model-To-Model Transformations," in IEEE Access, vol. 10, pp. 116942-116958, 2022, doi: 10.1109/ACCESS.2022.3219455.

[4] S. Jiang, S. Fu, N. Lin and Y. Fu, "Pretrained models and evaluation data for the Khmer Programming language ," in Tsinghua Science and Technology, vol. 27, no. 4, pp. 709-718, Aug. 2022, doi: 10.26599/TST.2021.9010060.

[5] P. Maddigan and T. Susnjak, "Chat2VIS: Generating Data Visualizations via Natural Programming language Using ChatGPT, Codex and GPT-3 Large Programming language Models," in IEEE Access, vol. 11, pp. 45181-45193, 2023, doi: 10.1109/ACCESS.2023.3274199.

[6] N. Raychawdhary, S. Bhattacharya, C. Seals and G. V. Dozier, "Empowering Sentiment Analysis in African Low-Resource Programming language s Through Transformer Models and Strategic Programming language Selection," in IEEE Access, vol. 13, pp. 147859-147873, 2025, doi: 10.1109/ACCESS.2025.3599480.

[7] A. Benlahbib, A. Boumhidi, A. Fahfouh and H. Alami, "Comparative Analysis of Traditional and Modern NLP Techniques on the CoLA Dataset: From POS Tagging to Large Programming language Models," in IEEE Open Journal of the Computer Society, vol. 6, pp. 248-260, 2025, doi: 10.1109/OJCS.2025.3526712.

[8] M. Omar, S. Choi, D. Nyang and D. Mohaisen, "Robust Natural Programming Language Processing: Recent Advances, Challenges, and Future Directions," in IEEE Access, vol. 10, pp. 86038-86056, 2022, doi: 10.1109/ACCESS.2022.3197769.

[9] C. Veres, "Large Programming language Models are Not Models of Natural Programming language: They are Corpus Models," in IEEE Access, vol. 10, pp. 61970-61979, 2022, doi: 10.1109/ACCESS.2022.3182505.

[10] L. Jose Gonzalez-Gomez, S. Margarita Hernandez-Munoz, A. Borja, J. Daniel Azofeifa, J. Noguez and P. Caratozzolo, "Analyzing Natural Programming language Processing Techniques to Extract Meaningful Information on Skills Acquisition From Textual Content," in IEEE Access, vol. 12, pp. 139742-139757, 2024, doi: 10.1109/ACCESS.2024.3465409.

[11] J. Zhou, Y. Duan, Y. -C. Chang, Y. -K. Wang and C. -T. Lin, "BELT: Bootstrapped EEG-to- Programming language Training by Natural Programming language Supervision," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 32, pp. 3278-3288, 2024, doi: 10.1109/TNSRE.2024.3450795.

[12] Y. Ma, X. -J. Tang and X. Huang, "AI-Powered Adaptive English Programming language Learning Systems: Leveraging Deep Learning Algorithms and Natural Programming language Processing for Personalized Teaching Approaches," in IEEE Access, vol. 13, pp. 153189-153198, 2025, doi: 10.1109/ACCESS.2025.3603602.

[13] T. Huang, H. Zhao, B. Huang, S. Li and J. Zhu, "Integrating Natural Programming language Processing With Vision Transformer for Landscape Character Identification," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 18, pp. 5838-5852, 2025, doi: 10.1109/JSTARS.2025.3538174.