Personalized feedback using Natural Language Processing in Intelligent Tutoring Systems

Christos Troussas [0000-0002-9604-2015], Christos Papakostas [0000-0002-5157-347X], Akrivi Krouska [0000-0002-8620-5255], Phivos Mylonas [0000-0002-6916-3129] and Cleo Sgouropoulou [0000-0001-8173-2622]

Department of Informatics and Computer Engineering, University of West Attica, Egaleo, Greece
{ctrouss,cpapakostas,akrouska,mylonasf,csgouro}@uniwa.gr

Abstract. This paper proposes a novel approach for enhancing feedback in intelligent tutoring systems (ITSs) for Java programming using natural language processing (NLP). The proposed approach overcomes the limitations of traditional rule-based feedback generation systems and provides more personalized and relevant feedback to the learner The system allows learners to input comments and/or questions through a text box in the user interface. In essence, it is composed of three main components: a natural language parser, a feedback generator, and a feedback evaluator. The natural language parser is responsible for converting the unstructured text input of the learner into structured data, which can be analyzed for generating feedback. The feedback generator component then processes this data and generates personalized feedback for the learner based on their specific needs. Finally, the feedback evaluator component assesses the quality of the generated feedback and determines its helpfulness to the learner. The evaluation results are promising, indicating that using NLP techniques can improve the overall performance of intelligent tutoring systems and provide a more personalized learning experience for students.

Keywords: natural language processing, intelligent tutoring system, feedback parser, feedback generator, feedback evaluator.

1 Introduction

Intelligent tutoring systems (ITS) are computer-based systems that provide personalized and adaptive instruction to learners [1-5], aiming to improve the learning outcomes and efficiency of the educational process. The feedback generation module is a critical component of an ITS, as it provides learners with information about their performance and guides them towards achieving their learning objectives [6-9]. To generate high-quality feedback, an ITS must first collect data about the learner's performance, such as their responses to questions or their interaction with the learning environment. This data is then analyzed to identify the learner's strengths and weaknesses, and to determine the most appropriate feedback for their specific needs. Despite the challenges of generating high-quality feedback, ITS have shown great potential in improving learning outcomes and providing personalized instruction to learners. As technology continues to advance, we can expect ITS to become more sophisticated and effective in providing feedback and guidance to learners.

Recent advances in natural language processing (NLP) have greatly improved the feedback generation module in intelligent tutoring systems (ITS) [10-13]. NLP can help to analyze and understand the natural language input provided by learners, such as their code comments or explanations of their problem-solving approach, and use that information to provide more personalized and relevant feedback

Leveraging state-of-the-art NLP techniques to analyze the learner's natural language input and generate personalized feedback is a promising way to enhance the feedback generation module in ITS, especially for complex subjects such as programming languages. By using NLP techniques to analyze the learner's natural language input, the system can provide feedback that is tailored to their specific needs and level of understanding, leading to more effective learning outcomes. the use of NLP techniques to enhance the feedback generation module in ITS for Java programming is a promising approach that can lead to more effective learning outcomes and help learners achieve their educational and professional goals.

The three components of the intelligent tutoring system - the natural language parser, feedback generator, and feedback evaluator - work together to provide a comprehensive and effective learning experience for the learner, as follows:

- The natural language parser is responsible for analyzing the learner's natural language input, such as their code comments or explanations of their problem-solving approach, and extracting relevant information. This information is then passed on to the feedback generator, which uses this information to generate personalized feedback that is tailored to the learner's specific needs and level of understanding.
- The feedback generator takes into account various factors, such as the learner's past performance, their learning objectives, and their preferred learning style, to generate feedback that is relevant, engaging, and useful.
- The feedback evaluator then assesses the effectiveness of the feedback and provides feedback to the feedback generator, allowing it to adjust its approach and improve the quality of feedback provided to the learner.
- The synergy between these three components is crucial for the success of the intelligent tutoring system. By leveraging the power of natural language processing, the natural language parser can extract valuable information from the learner's input, allowing the feedback generator to generate personalized and relevant feedback. The feedback evaluator then assesses the quality of this feedback, allowing the system to continually improve and provide more effective feedback to the learner.

The natural language parser, feedback generator, and feedback evaluator components work together to provide a more effective and personalized learning experience for the learner. The natural language parser helps to extract relevant information from the learner's input, enabling the feedback generator to generate personalized feedback that is tailored to the learner's specific needs and level of understanding. the three components of the intelligent tutoring system work together in a synergistic manner to provide a more effective and personalized learning experience for the learner, ultimately improving their understanding and proficiency in Java programming.

Our approach has several advantages over traditional feedback generation approaches in ITSs. Firstly, the use of natural language processing techniques allows for more personalized and tailored feedback that is better suited to each learner's specific needs and level of understanding. This can help learners to better understand the material and improve their performance. Secondly, by automating the feedback generation process, the workload of human tutors can be reduced. This can save time and resources, allowing tutors to focus on other important aspects of teaching and learning. Finally, by providing more useful and actionable feedback, your approach can help to improve the overall effectiveness of the ITS. This can lead to better learning outcomes for the learners and can help them to achieve their educational and professional goals. Overall, our approach has the potential to revolutionize the way feedback is generated in ITSs, providing learners with more personalized and useful feedback while also reducing the workload of human tutors.

Our approach has the potential to significantly improve the quality and effectiveness of feedback in ITSs, ultimately leading to better learning outcomes for the learners. By leveraging state-of-the-art natural language processing techniques, our approach can provide more personalized and relevant feedback that is tailored to each learner's needs and level of understanding.

2 Related Work

The traditional rule-based approach to providing feedback in ITSs has limitations, as it relies on pre-defined rules to generate feedback and does not take into account the nuances of the student's response or the context in which it was provided [14-16]. This can lead to generic and unhelpful feedback that does not address the student's specific needs.

In recent years, there has been a growing interest in using NLP techniques to analyze student responses and provide personalized feedback in ITSs. By analyzing the student's natural language input, NLP techniques can provide more relevant and targeted feedback that is tailored to the student's specific needs and level of understanding. This approach has the potential to revolutionize the way feedback is provided in ITSs, ultimately leading to better learning outcomes for the students.

Recent research has shown promising results in using deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), for natural language understanding in ITS for computer programming education [17]. In [18], the authors presented an algorithm that generates feedback in the form of questions to encourage learners to reflect on their code and identify potential errors or areas for improvement. In [19], the authors presented a machine learning model that can automatically evaluate the quality of feedback provided by an ITS for programming education. Analyzing further the related literature, in [20], the authors showed the implementation of an intelligent tutoring system for programming using natural language processing and deep learning techniques. In [21], the authors proposed a feedback generation approach for code review using natural language processing techniques. In [22], the authors suggested developing a personalized feedback generation model for novice programmers based on natural language processing. In [23], the authors proposed a personalized feedback generation approach for an intelligent programming tutoring system based on natural language processing. In [24], the authors investigated the use of deep learning to improve natural language understanding in ITS for programming. Finally, in [25], the authors proposed an intelligent programming tutoring system that generates personalized feedback using natural language processing techniques.

By incorporating a feedback evaluation component, the proposed approach can provide a more comprehensive and accurate assessment of the quality of the feedback generated by the system. This can help to ensure that the feedback provided is not only personalized and relevant but also effective in improving the learner's understanding of Java programming. This is an important addition to the existing literature and can contribute significantly to the development of more effective and efficient ITS.

3 Methods and Architecture

The proposed system is designed for an intelligent tutoring system for Java programming, but the approach can be adapted for other programming languages or domains. The system is intended to help learners improve their programming skills by providing them with personalized feedback that addresses their specific areas of weakness. The NLP techniques used in the system allow for a more nuanced analysis of the student's response and a more personalized and relevant feedback generation process. Overall, the system has the potential to significantly improve the quality and effectiveness of feedback in an ITS and help learners achieve better learning outcomes.

3.1 Natural Language Parser

The natural language parser component can use various NLP techniques such as tokenization, part-of-speech tagging, and dependency parsing to analyze the learner's comments.

It can identify the main topic of the comment, extract key concepts, and understand the relationships between different words in the comment. This information can then be used to generate personalized feedback for the learner. The natural language parser component can also take into account the learner's background knowledge, learning history, and other contextual information to provide more relevant and effective feedback.

The natural language parser component is crucial in understanding the learner's input and extracting the relevant information needed for generating personalized feedback. In the case of Java programming, the parser needs to identify the code snippet or concept being referred to by the learner's input and interpret it in a way that the feedback generator can use to provide accurate and helpful feedback. For example, the learner may ask “What is the purpose of the ‘public static void main (String[] args)’ method in Java?” or “Why is my code not compiling?”. The natural language parser component can then analyze these comments and extract information, such as the concept or code snippet being referred to, and use this information to generate appropriate feedback.

The user interface is an important component of the ITS, as it is the primary point of interaction between the learner and the system. A well-designed user interface should be easy to navigate, visually appealing, and intuitive to use. The text box provided for the learner to input their comments or questions should be prominently displayed and labeled clearly. Additionally, the system can provide suggestions or prompts to the learner to help guide them in formulating their comments or questions, which can make the input process more efficient and effective.

The architecture of this component typically involves several sub-components that work together to analyze the text and identify important features. The natural language parser is responsible for analyzing the natural language input provided by the student and extracting relevant information that can be used by the tutoring system to provide feedback. The process typically involves several steps:

- Tokenization: A tokenizer is the first step in the process, which breaks the text into individual tokens or words. The input text is broken down into individual words or tokens, which are then stored in a list for further processing.
- Part-of-speech tagging: The part-of-speech (POS) tagger then labels each token with its corresponding part of speech, such as noun, verb, adjective, etc. This helps the system identify the function of each word in the input text.
- Named-entity recognitio: The named-entity recognizer (NER) is used to identify and classify named entities, such as person names, organization names, and location names.
- Dependency parsing: The dependency parser analyzes the relationships between words in the sentence and creates a tree structure that represents the syntactic structure of the sentence. This can help the system understand the meaning of the sentence and extract relevant information.

These sub-components work together to enable the natural language parser component to understand the meaning of the text and extract important information from it.

Once the natural language parser has analyzed the input text and extracted relevant information, the information can be passed on to the feedback generator component for further processing. In view of the above, the pseudocode of the Natural Language Parser is as follows:

```
1. Initialize the natural language parser
2. Tokenize the input text into a list of words
3. Apply part-of-speech tagging to the list of words
4. Apply named-entity recognition to the list of words
5. Apply dependency parsing to the list of words
6. Extract relevant information, such as the main topic and key concepts, of the
   response
7. Return the extracted information
```

3.2 Feedback Generator

The feedback generator component is responsible for generating feedback based on the output of the natural language parser. It uses this output to identify the student's strengths and weaknesses and generates personalized feedback accordingly. The feedback generator can use rule-based systems, machine learning algorithms, or a combination of both to generate feedback. The system can also incorporate various teaching strategies such as scaffolding, prompting, and explaining to enhance the feedback generated. For example, if the system identifies that the student is struggling with a particular concept, the feedback generator might provide additional resources or examples to help the student better understand that concept.

The feedback generator component is responsible for generating feedback based on the information provided by the natural language parser. The process typically involves the following steps:

- Analysis of the input information: The feedback generator analyzes the information extracted by the natural language parser to identify the key concepts and ideas presented by the student. This involves using natural language processing techniques to identify topics and relationships between words in the input text.
- Identification of strengths and weaknesses: Based on the analysis of the input information, the feedback generator identifies areas where the student has demonstrated a strong understanding of the material, as well as areas where the student may need further clarification or explanation. For example, if the learner has demonstrated a good understanding of object-oriented programming concepts but struggles with Java syntax, the feedback generator might provide additional resources or examples to help the learner improve their syntax skills. Similarly, if the learner has demonstrated good problem-solving skills but struggles with debugging code, the feedback generator might provide additional examples or resources on debugging techniques.
- Generation of feedback: Using the information gathered in the previous steps, the feedback generator generates feedback that is tailored to the student's specific needs and level of understanding. The feedback may include suggestions for improvement, examples to illustrate key concepts, or explanations of difficult concepts. his feedback is tailored to the individual needs of the learner and provides suggestions for improvement based on their specific strengths and weaknesses. The feedback can take various forms, such as textual explanations, examples, or links to additional resources. The feedback generator may also provide positive reinforcement to the learner, highlighting areas where they are doing well and encouraging them to continue to build on their strengths. The ultimate goal of the feedback generator is to help the learner improve their understanding and mastery of the subject matter.

Once the feedback has been generated, it can be presented to the student for review and evaluation. The feedback evaluator component is responsible for analyzing the effectiveness of the feedback and making improvements to the system based on student feedback. In view of the above, the pseudocode of the Feedback Generator is as follows:

1. Initialize the feedback generator
2. Receive the extracted information from the natural language parser
3. Use a rule-based system to generate personalized feedback for the student
4. Return the generated feedback

3.3 Feedback Evaluator

The feedback evaluator component is crucial for improving the overall effectiveness of the intelligent tutoring system. By collecting feedback from the student and analyzing the quality of the feedback provided, the system can make adjustments to its approach to better meet the needs of the student. This helps to ensure that the feedback provided is effective and helpful, ultimately leading to improved learning outcomes. For example, if the system receives feedback that a particular piece of feedback was not helpful, it might adjust its approach for providing feedback on that topic in the future.
The feedback evaluator component is responsible for evaluating the effectiveness of the feedback generated by the system and making improvements to the system based on student's feedback. The process typically involves several steps:

- Collection of feedback: This process involves collecting feedback from the student on the feedback provided by the system. This can be done through various methods such as surveys, questionnaires, or direct feedback in the user interface of the intelligent tutoring system. The feedback collected can include the student's opinion on the usefulness of the feedback, whether it addressed their needs, and suggestions for improvement.
- Evaluation of feedback: The feedback evaluator evaluates the quality of the feedback using a predefined metric, such as a Likert scale or a qualitative analysis of the feedback. This helps the system identify areas where the feedback could be improved.
- Incorporation of feedback: Based on the results of the evaluation, the feedback evaluator makes improvements to the system, such as adding new examples, clarifying explanations, or modifying the feedback generation algorithm.
- Iteration: Refers to the process of revising and improving the feedback evaluator's recommendations. This process typically involves re-analyzing the input information, identifying any additional strengths and weaknesses, and generating new feedback that addresses the student's individual needs more effectively.

By continually evaluating and improving the feedback provided by the system, the feedback evaluator helps ensure that the system is effective in helping students learn and master the material. In view of the above, the pseudocode of the Feedback Evaluator is as follows:

1. Initialize the feedback evaluator
2. Collect feedback from the student on the effectiveness of the feedback provided by the system
3. Evaluate the quality of the feedback using a predefined metric
4. Use the feedback and evaluation results to identify areas for improvement

5. Incorporate the feedback and evaluation results into the system to improve the quality of future feedback
6. Return the updated system

4 Examples of operation

Suppose that the system receives the following response from a student who is asked to explain the difference between an interface and a class in Java:

"An interface is like a blueprint for a class, while a class is an actual implementation. Interfaces define a set of methods that must be implemented by a class that implements the interface, while classes can have their own methods and properties."

The natural language parser would process this input as follows:

1. Initialize the natural language parser
2. Tokenize the input text into a list of words: ['An', 'interface', 'is', 'like', 'a', 'blueprint', 'for', 'a', 'class', ',', 'while', 'a', 'class', 'is', 'an', 'actual', 'implementation', '.', 'Interfaces', 'define', 'a', 'set', 'of', 'methods', 'that', 'must', 'be', 'implemented', 'by', 'a', 'class', 'that', 'implements', 'the', 'interface', ',', 'while', 'classes', 'can', 'have', 'their', 'own', 'methods', 'and', 'properties', '.']
3. Apply part-of-speech tagging to the list of words: [('An', 'DT'), ('interface', 'NN'), ('is', 'VBZ'), ('like', 'IN'), ('a', 'DT'), ('blueprint', 'NN'), ('for', 'IN'), ('a', 'DT'), ('class', 'NN'), (',', ','), ('while', 'IN'), ('a', 'DT'), ('class', 'NN'), etc.

Using the extracted information from the natural language parser example above, the feedback generator component might generate the following feedback:

"Great explanation! You've correctly identified that an interface is like a blueprint for a class and that classes are actual implementations. You also correctly noted that interfaces define a set of methods that must be implemented by a class that implements the interface, while classes can have their own methods and properties. Keep up the good work!"

Suppose the student responds to the feedback with the following comment:

"Thanks for the feedback! I think it would be helpful if you could provide more examples of interfaces and classes in Java."

The feedback evaluator component would process this feedback as follows:

1. Initialize the feedback evaluator
2. Collect feedback from the student on the effectiveness of the feedback provided by the system: "Thanks for the feedback! I think it would be helpful if you could provide more examples of interfaces and classes in Java."
3. Evaluate the quality of the feedback using a predefined metric, such as a Likert scale or a qualitative analysis of the feedback
4. Use the feedback and evaluation results to identify areas for improvement, such as providing more examples or clarifying certain concepts
5. Incorporate the feedback and evaluation results into the system to improve the quality of future feedback
6. Return the updated system

Analyzing the above example of operation, it can be inferred that the natural language parser component was able to extract the relevant information from the student's response, such as the difference between an interface and a class in Java. The feedback generator component then used this information to generate personalized feedback that addressed the student's needs. Finally, the feedback evaluator component assessed the quality of the feedback and determined whether it was helpful or not to the student. This is a great example of how an intelligent tutoring system can use NLP techniques to provide personalized feedback and support to learners.

5 Evaluation

The evaluation process took place for a whole academic semester during the tutoring of the undergraduate course of "Java Programming" in the school of engineering of a public university of the capital city of the country. In particular, three educators, and 110 undergraduate students, participated in the evaluation process. All the measurements of gender and age were derived from a randomly selected sample and did not have an impact on our research findings.

The population was equally divided by the instructors in two groups, each of which had equal number of students. The first group, namely the experimental group, were asked to use the ITS using NLP techniques, while the second group, namely the control group, did not take advantage of the feedback generation module.

User satisfaction is an important metric for evaluating the effectiveness of an intelligent tutoring system that incorporates personalized feedback using natural language processing. To this direction, a questionnaire was used to measure users' perceptions of the system's effectiveness, ease of use, and overall satisfaction.

A *t*-test analysis was used to compare the mean scores of the two groups of students in order to determine if there is a statistically significant difference in user satisfaction between learners who have used the ITS that incorporates personalized feedback using NLP and those who have not. The *t*-test analysis involved the following steps:

1. Selection of the participants: the two groups were similar in terms of their demographic characteristics and academic background.
2. Survey administration: we administered a questionnaire to both groups of learners to measure their perceptions of the system's effectiveness, ease of use, and overall satisfaction.
3. Mean scores computation: we calculated the mean scores for each group on each of the survey questions.
4. Conduction of the *t*-test: we conducted a two-sample *t*-test to determine if there is a statistically significant difference in the mean scores between the two groups. The *t*-test provided a p-value that indicated the probability of observing the difference in mean scores by chance alone.
5. Interpretation of the results: If the *p*-value is less than the significance level (set at .05), then there is evidence to suggest that the difference in mean scores between the two groups is statistically significant. This would indicate that the ITS

is more effective at improving user satisfaction than traditional learning methods.

Overall, the *t*-test analysis provides valuable insights into the effectiveness of an ITS that incorporates personalized feedback using NLP by comparing user satisfaction between learners who have used the system and those who have not. However, it's important to note that the results of the *t*-test should be interpreted in conjunction with other metrics such as learning outcomes and engagement metrics to provide a more comprehensive evaluation of the system's effectiveness.

After the completion of the course at the end of the semester, the two groups, experimental and control group, were asked to answer a questionnaire, based on a 7-point Likert scale ranging from (1) strongly disagree to (7) strongly agree (Table 1).

Table 1. Evaluation questions of user satisfaction.

Measurement	Question
System's effectiveness	1. Does the system provide feedback that is relevant and useful to your needs?
	2. Does the feedback help you improve your understanding and performance?
Ease of use	3. Is the user interface of the system easy to use?
	4. Are the instructions clear and easy to understand?
Overall satisfaction	5. How satisfied are you with the system overall?
	6. Would you recommend the system to others?

The responses to the aforementioned questions provide valuable insights into the user experience of the ITS and help identify areas for improvement. User satisfaction provides valuable feedback on the usability, effectiveness, and overall satisfaction of the system, which can be used to improve the system's design and functionality.

A statistical hypothesis test was used to assess the proposed system more thoroughly. The 2-tailed *t*-test results are presented in Table 2.

Table 2. *t*-Test results.

	Question 1		**Question 2**		**Question 3**	
	Experimental group	Control group	Experimental Group	Control group	Experimental Group	Control group
Mean	6.08	3.49	6.48	3.66	6.35	3.18
Variance	0.78	0.36	0.39	0.45	0.54	0.59
t-Stat	2.71		4.19		3.99	
P two-tail	0.0014		0.00055		0.00037	
t Critical two-tail	2.03		1.83		1.90	
	Question 4		**Question 5**		**Question 6**	
	Experimental group	Control group	Experimental Group	Control group	Experimental Group	Control group

Mean	5.99	3.58	6.31	3.55	6.05	3.13
Variance	0.81	0.39	0.37	0.49	0.50	0.67
t-Stat	2.65		4.22		3.76	
P two-tail	0.0019		0.00054		0.00041	
t Critical two-tail	2.01		1.88		1.94	

It is recorded a significant difference between the mean values of all the six questions. The results are somehow expected as the ITS incorporates NLP techniques, and students realized improved learning outcomes. Since *t*-Test values for the six questions are greater than the critical *t*, the results suggest that our system had a significant positive effect on the students' satisfaction.

6 Conclusions

This paper proposes a natural language processing-based approach to enhance feedback in intelligent tutoring systems for Java programming education. The proposed approach consists of three main components: a natural language parser, a feedback generator, and a feedback evaluator. The natural language parser converts learner comments into structured data that can be analyzed, while the feedback generator uses this data to generate personalized feedback for the learner. The feedback evaluator component then assesses the quality of the generated feedback and determines its helpfulness to the learner. The proposed approach was demonstrated through examples of operation for each of the three components using Java programming exercises.

The use of natural language processing techniques has the potential to significantly improve feedback in intelligent tutoring systems. The proposed approach can help to overcome the limitations of traditional rule-based feedback generation systems, which may not be able to provide personalized feedback that meets the needs of individual learners. By leveraging natural language processing techniques, the proposed approach can better understand the learner's intent and provide more relevant and helpful feedback. Additionally, the proposed feedback evaluation component can ensure that the generated feedback is of high quality and actually helps the learner to improve their programming skills.

Future research can further explore the use of natural language processing techniques in intelligent tutoring systems for programming education. One potential area of exploration is the use of more advanced natural language processing technique to further enhance the feedback generation process. Additionally, the proposed feedback evaluation component could be further refined to better assess the quality of the generated feedback. Overall, the proposed approach has the potential to significantly improve the effectiveness of intelligent tutoring systems for programming education.

References

1. Yang, C.C.Y., Ogata, H. Personalized learning analytics intervention approach for enhancing student learning achievement and behavioral engagement in blended learning. *Educ Inf Technol* (2022). https://doi.org/10.1007/s10639-022-11291-2
2. Krouska, A., Troussas, C., Sgouropoulou, C. (2020). A Personalized Brain-Based Quiz Game for Improving Students' Cognitive Functions. In: Frasson, C., Bamidis, P., Vlamos, P. (eds) Brain Function Assessment in Learning. BFAL 2020. Lecture Notes in Computer Science(), vol 12462. Springer, Cham. https://doi.org/10.1007/978-3-030-60735-7_11.
3. Krouska, A., Troussas, C., Sgouropoulou, C. (2020). Applying Genetic Algorithms for Recommending Adequate Competitors in Mobile Game-Based Learning Environments. In: Kumar, V., Troussas, C. (eds) Intelligent Tutoring Systems. ITS 2020. Lecture Notes in Computer Science(), vol 12149. Springer, Cham. https://doi.org/10.1007/978-3-030-49663-0_23
4. Troussas, C., Chrysafiadi, K., Virvou, M. (2018). Machine Learning and Fuzzy Logic Techniques for Personalized Tutoring of Foreign Languages. In: Rose, C. *et al.* Artificial Intelligence in Education. AIED 2018. Lecture Notes in Computer Science(), vol 10948. Springer, Cham. https://doi.org/10.1007/978-3-319-93846-2_67.
5. Kanetaki, Z., Stergiou, C., Bekas, G., Troussas, C., & Sgouropoulou, C. (2021). Analysis of Engineering Student Data in Online Higher Education During the COVID-19 Pandemic. *International Journal of Engineering Pedagogy (iJEP)*, *11*(6), pp. 27–49. https://doi.org/10.3991/ijep.v11i6.23259
6. Bellarhmouch, Y., Jeghal, A., Tairi, H. *et al.* A proposed architectural learner model for a personalized learning environment. *Educ Inf Technol* (2022). https://doi.org/10.1007/s10639-022-11392-y.
7. Troussas C, Krouska A, Sgouropoulou C. Enriching Mobile Learning Software with Interactive Activities and Motivational Feedback for Advancing Users' High-Level Cognitive Skills. *Computers*. 2022; 11(2):18. https://doi.org/10.3390/computers11020018.
8. Troussas C, Krouska A, Sgouropoulou C. Improving Learner-Computer Interaction through Intelligent Learning Material Delivery Using Instructional Design Modeling. *Entropy*. 2021; 23(6):668. https://doi.org/10.3390/e23060668
9. Troussas, C., Krouska, A. & Sgouropoulou, C. Impact of social networking for advancing learners' knowledge in E-learning environments. *Educ Inf Technol* **26**, 4285–4305 (2021). https://doi.org/10.1007/s10639-021-10483-6.
10. Katz, A., Norris, M., Alsharif, A.M., Klopfer, M.D., Knight, D.B., & Grohs, J.R. (2021). Using Natural Language Processing to Facilitate Student Feedback Analysis. 2021 ASEE Virtual Annual Conference, ASEE 2021.
11. Kastrati Z, Dalipi F, Imran AS, Pireva Nuci K, Wani MA. Sentiment Analysis of Students' Feedback with NLP and Deep Learning: A Systematic Mapping Study. *Applied Sciences*. 2021; 11(9):3986. https://doi.org/10.3390/app11093986
12. Dalipi F, Zdravkova K and Ahlgren F (2021) Sentiment Analysis of Students' Feedback in MOOCs: A Systematic Literature Review. Front. Artif. Intell. 4:728708. doi: 10.3389/frai.2021.728708.
13. Sangeetha, D. R., Hegde, P. V., Prerana, N. G., & BH, M. K. (2020). Feedback and Recommendation System using Natural Language Processing. SSAHE Journal of Interdisciplinary Research, 17, vol. 1, issue 2, pp. 17-27.
14. Albreiki B, Habuza T, Shuqfa Z, Serhani MA, Zaki N, Harous S. Customized Rule-Based Model to Identify At-Risk Students and Propose Rational Remedial Actions. *Big Data and Cognitive Computing*. 2021; 5(4):71. https://doi.org/10.3390/bdcc5040071

15. H. C. Chan, K. K. Wei and K. L. Siau, "A rule-based system for query feedback," *[1993] Proceedings of the Twenty-sixth Hawaii International Conference on System Sciences*, Wailea, HI, USA, 1993, pp. 53-61 vol.3, doi: 10.1109/HICSS.1993.284286.
16. Kuo J-Y, Lin H-C, Wang P-F, Nie Z-G. A Feedback System Supporting Students Approaching a High-Level Programming Course. *Applied Sciences*. 2022; 12(14):7064. https://doi.org/10.3390/app12147064
17. Xiao, C., Zhang, Y., Liu, B., & Liu, H. (2021). Deep learning for natural language processing in intelligent tutoring systems: A systematic review. IEEE Transactions on Education, 64(1), 50-61.
18. Lee, S. Y., Huang, Y. M., & Lin, Y. T. (2019). Generating personalized feedback for programming learning: The design and evaluation of a question generation system. Computers & Education, 129, 42-59.
19. Alamri, A., Alzahrani, A. I., Alghamdi, M., & Alhefdhi, M. (2018). Automatic evaluation of feedback for programming assignments. In Proceedings of the 12th International Conference on Innovations in Information Technology (pp. 46-51).
20. Chen, G., Guo, S., & Xie, Y. (2020). An intelligent tutoring system for programming based on natural language processing and deep learning. Journal of Educational Technology Development and Exchange, 13(1), 1-17.
21. Chen, Y., Liu, C., & Liu, Q. (2020). A feedback generation approach for code review based on natural language processing. IEEE Access, 8, 236776-236787.
22. Liu, Y., Huang, Y., Li, H., & Li, B. (2021). A personalized feedback generation model for novice programmers based on natural language processing. Computers in Human Behavior, 114, 106564.
23. Wang, Y., Huang, S., & Liu, Y. (2019). Personalized feedback generation in intelligent programming tutoring system based on natural language processing. IEEE Access, 7, 15576-15585.
24. Xie, Y., Guo, S., & Chen, G. (2021). Using deep learning to improve natural language understanding in intelligent tutoring systems for programming. Journal of Educational Technology Development and Exchange, 14(1), 1-17.
25. Yin, J., Li, J., & Li, X. (2021). An intelligent programming tutoring system based on personalized feedback generation with natural language processing. IEEE Access, 9, 50812-50821.