International Journal of Pedagogics

VOLUME Vol.05 Issue06 2025
PAGE NO. 185-189
DOI 10.37547/ijp/Volume05Issue06-51

The Development Of Automated Creating Test Questions Using Artificial Intelligence: Knowledge Estimation

Yuldashev Nodirbek Abdumannob o'g'li
Andijan State Technical Institute, Uzbekistan

Received: 14 April 2025; **Accepted:** 15 May 2025; **Published:** 19 June 2025

Abstract: The main point of the thesis is the generation of questions from text using automation. Different methods of artificial intelligence and natural language processing are explained, along with the potential for adaptive learning in environments where questions are generated automatically. The student model is a compilation of ongoing data about the student that determines which skill to work on next. The report may contain details regarding the student's proficiency in certain skills, as well as their level of enjoyment and motivation. During fact practicing, the student model is utilized to predict how likely it is that a student knows a specific fact. The paragraphs below outline the most frequently used student models for factual knowledge.

Keywords: Question generation, adaptive practice, knowledge representation, knowledge extraction, learning, multiple choice questions.

Introduction

The one-parameter logistic model is the fundamental model of student behavior in the item response theory. This model utilizes a logistic function with a parameter c that represents the difficulty level of the fact. The formula gives the probability of a student with knowledge k knowing a fact with difficulty d.

$$P(k,c) = \frac{1}{1+e^{-(k-c)}}$$

Formula 3.1

The logistic function (Formula 3.2) is appropriate for modeling probabilities due to its range of 0 to 1 for all inputs. When dealing with multiple choice questions, we must modify the formula slightly to accommodate for guessing. When there are n options available, the likelihood of making a correct guess is $\frac{1}{n}$.

$$P(k,c,n) = \frac{1}{n} + \left(1-\frac{1}{n}\right) * \frac{1}{1+e^{-(k-c)}}$$

Formula 3.2

With sufficient data, one can estimate model parameters using a maximum likelihood estimation algorithm. Nevertheless, as we employ questions that are created, there is no information available regarding students' responses to them. This is why it's important to have approximate judgments of complexity. The student's level of understanding is determined by analyzing the questions they have previously answered and the challenges they faced, following the maximum likelihood principle, where the knowledge level chosen maximizes the likelihood of the student's progress. Even though it is unsolvable analytically, quick numerical techniques like the Newton-Raphson method can be employed.

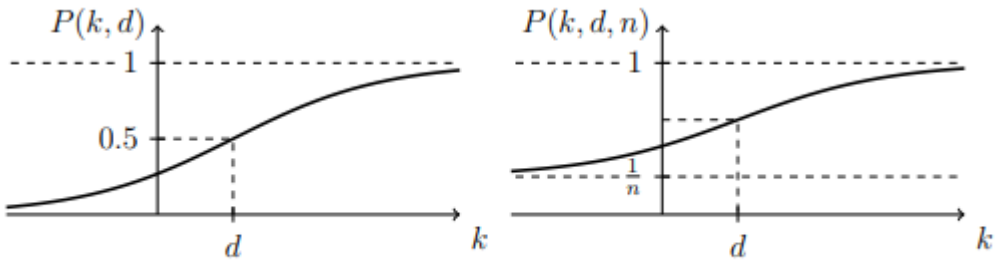

Figure 4.2: Logistic model without guessing (left) and with guessing (right)

- **Performance factors analysis:**

Performance factors analysis is also similar to the logistic model and enables to capture learning. According to this model, the probability that a student answers correctly to a question depends on the difficulty c of the question and number of correct (a) and incorrect (b) answers of the student to this question in the past. The exact formula is:

$$P(a,b,c) = \frac{1}{1 + e^{-k}} \; where \; k = \alpha * a + \beta * b - c$$

Formula 3.3

This model's primary limitation is disregarding the sequence in which answers are provided. For instance, responding incorrectly 10 times followed by 10 correct answers yields the same prediction as alternating correct and incorrect responses. While the former suggests learning, the latter indicates the student has not yet learned the material. Performance factors analysis can be integrated with the new model by updating the knowledge parameter k after each answer, following the rule 3.3 to consider the order of answers.

Target Difficulty Adjustment

As previously stated in the chapter introduction, the key to effective learning is giving the student appropriately challenging questions. Even though the optimal success rate can differ among individuals and situations, it is commonly simplified as a fixed percentage, such as 75%.

Tests conducted on the internet-based adaptive learning platform for geography facts indicate that the best success rate could be between 65 to 70 %. The outcome came from students' self-reports after answering questions, rating the difficulty as "too easy," "appropriate," or "too difficult". Each student was placed in one of 10 groups with varying target probabilities of correct answers, ranging from 50 to 95%.

It was observed that the success rate is typically greater than the intended probability of a correct answer. In order to get closer to the targeted success rate, a new algorithm was suggested for adjusting the difficulty of the target dynamically. It is built on the concept of a proportional controller. If the current rate of success is below the desired probability of a correct answer, the desired probability will be raised in proportion to the "error." If the current success rate increases, the desired probability is reduced accordingly. The precise modification function can be seen in Formula 3.3.

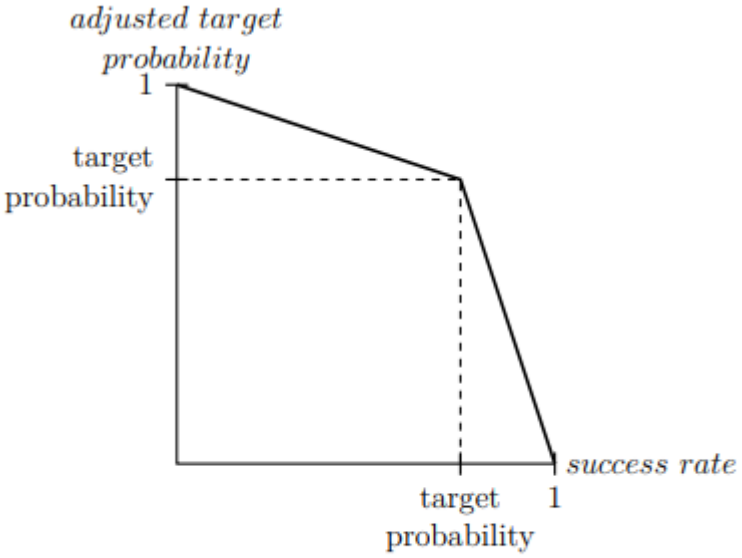

Figure 4.3: Function describing adjustment of target probability

This dynamic adjustment of target probability proved to be useful – it increased by 5 % the set of people who reported that the difficulty is “appropriate”.

Question Selection:

In earlier parts, we talked about determining students' understanding and changing question difficulty levels accordingly. By utilizing these two elements, we can determine the ideal level of difficulty for the question. Nevertheless, various factors need to be taken into account when choosing the most beneficial question for the student, with difficulty being just one aspect to consider. Let's examine the key criteria.

The question must be pertinent to the subject being studied. In addition to the objective importance that does not depend on the student or practice session, there is also subjective importance - the concept most relevant to the student at the present moment. The subjective importance varies throughout the practice, with some parts of the article having been

practiced while others have not. Typically, the concept that is least familiar to the student is usually the most personally significant, as long as it is also important in terms of the subject matter.

Challenge - As previously mentioned, the ideal likelihood of a correct response is around 70 % and a technique for adjusting this target probability dynamically was also introduced in chapter 3.2. By utilizing a student model in chapter 3.1, we can predict the likelihood of a student answering a specific question accurately. The closer the estimated probability is to the target probability, the more preferred this question is.

The diversity of practiced concepts is particularly crucial, especially when it comes to variability. If a single idea is repeated multiple times in consecutive questions, it becomes dull quickly, impacting the importance (as students prefer to work on different ideas) and the challenge (since answering questions about the same idea is simpler). In addition to the important differences in concepts, having different types of questions can also be beneficial as it enhances enjoyment due to diversity. In Slepé mapy and DynaLearn, the diversity of question types is considered.

A typical strategy involves utilizing a suitable scoring function for every criterion under consideration, and then calculating the overall score by combining the individual scores in a linear manner. The highest total score determines which question is chosen and given to the student.

AQG Project Framework

AQG Project is a framework for question generation and adaptive practice. It is written in a modular way and decomposes the process into the following components:

- KnowledgeBuilder – takes an article as the input and extracts knowledge, building a RDF graph of facts contained in the article.
- ExercisesCreator – creates a set of questions (or possibly other types of exercises) using the built knowledge graph.
- ExercisesGrader – ranks generated exercises with respect to attributes such as difficulty or relevance to the original article.
- Practicer – controls the practice session itself, i.e. gives the student one exercise at a time based on the previous exercises and answers.

Each component (e.g. *KnowledgeBuilder*) has variable behavior (e.g. *KnowledgeBuilderBehavior*), which is a parametrised code, i.e. an implementation of the behavior (the code) and a set of parameters which the code uses. Component itself is just a wrapper around the behavior which performs common tasks, most importantly persistence management. At the beginning of each new session, 4-tuple of components with behaviors is chosen.

Figure 3.1 : Components and behaviors

The proposed modular design allows for easy integration and testing of new behaviors. Simple prototype behaviors based on heuristic approaches were implemented. More sophisticated behaviors will be the subject of future development.

Framework Except for *ExercisesCreator* and *ExercisesGrader*, the components does not communicate with each other directly, but via results (knowledge graphs and exercises) stored in the database. The communication and the flow of data is depicted. Individual processes are triggered by signals from the client.

Figure 3.2: Data flow diagram

Behaviors Selection

Each session has assigned a *performance* which is computed as a weighted sum of the user final rating and the ratio of exercises marked as invalid or irrelevant. As we know which behaviors participated in which sessions, we can compute the average performance of a 4-tuple of behaviors $p(x_1, x_2, x_3, x_4)$.

At the beginning of each new session, 4-tuple of behaviors (k_1, k_2, k_3, k_4) (k1 is knowledge builder, k_2 exercises creator, etc.) is chosen in a partially random way which prioritizes behaviors with better performance in the past. Ideally, the probability of a certain 4-tuple being selected would be proportional to its expected relative performance (relative to the expected performances of other 4-tuples). **Automated Question Generation(AQG) Project Framework**

AQG Project is a framework for question generation and adaptive practice. It is written in a modular way and decomposes the process into the following components:

- KnowledgeBuilder – takes an article as the input and extracts knowledge, building a RDF graph of facts contained in the article.
- ExercisesCreator – creates a set of questions (or possibly other types of exercises) using the built knowledge graph.
- ExercisesGrader – ranks generated exercises with respect to attributes such as difficulty or relevance to the original article.
- Practicer – controls the practice session itself, i.e. gives the student one exercise at a time based on the previous exercises and answers.

Each component (e.g. *KnowledgeBuilder*) has variable behavior (e.g. *KnowledgeBuilderBehavior*), which is a parametrised code, i.e. an implementation of the behavior (the code) and a set of parameters which the code uses. Component itself is just a wrapper around the behavior which performs common tasks, most importantly persistence management. At the beginning of each new session, 4-tuple of components with behaviors is chosen.

Figure 3.1 : Components and behaviors

The proposed modular design allows for easy integration and testing of new behaviors. Simple prototype behaviors based on heuristic approaches were implemented. More sophisticated behaviors will be the subject of future development. Framework Except for *ExercisesCreator* and *ExercisesGrader*, the components does not communicate with each other directly, but via results (knowledge graphs and exercises) stored in the database.

Figure 3.2: Data flow diagram

Behaviors Selection

Each session has assigned a *performance* which is computed as a weighted sum of the user final rating and the ratio of exercises marked as invalid or irrelevant. As we know which behaviors participated in which sessions, we can compute the average performance of a 4-tuple of behaviors $p(x_1, x_2, x_3, x_4)$.

At the beginning of each new session, 4-tuple of behaviors (k_1, k_2, k_3, k_4) (k1 is knowledge builder, k_2 exercises creator, etc.) is chosen in a partially random way which prioritizes behaviors with better performance in the past. Ideally, the probability of a certain 4-tuple being selected would be proportional to its expected relative performance (relative to the expected performances of other 4-tuples).

Exercises Creating

ExerciseCreator component uses built knowledge graph to generate a set of exercises. Exercises consist of presentation data and semantic information.

- presentation data Currently the only supported type of exercise is a multiple choice question.
- semantic information Semantic information is used for exercises grading. For this purpose, I decided to use a list of term pairs for which holds, that if the user confuses them, it can lead to an incorrect answer. For a multiple choice question with correct answer A and distractors *B, C* and *D*, the term pairs are *(A, B), (A, C) and (A, D).*

Deployment and Evaluation:

The project structure is described in Appendix A, the source code is available from its repository. AQG Project has been deployed at http://4200.localhost. In this chapter, we mention a few deployment details and then I analyze the feedback collected from the users.

CONCLUSION

In conclusion, this dissertation thesis has shed light on the immense potential of AI and NLP in automating the question generation process for learning management systems. By automating assessments, personalizing learning experiences, providing formative feedback, enriching content, and fostering critical thinking, automatic question generation can revolutionize the educational landscape. The findings and insights presented in this thesis contribute to the body of knowledge in the field and pave the way for further research and innovation in AI-powered question generation for LMS.

REFERENCES

N. Yuldashev “The role and utilizing of automated question generation into web based online examination systems”, in the Journal of Engineering and Technology (JET) ISSN(P): 2250-2394; ISSN(E): Applied Vol. 13, Issue 2, Dec 2023, 157-164 © TJPRC Pvt. Ltd.

M. Agarwal, R. Shah, and P. Mannem, “Automatic question generation using

discourse cues”, in Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational

Applications, Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 1–9, isbn: 9781937284039.

N. Bach and S. Badaskar, "A review of relation extraction", Literature review
for Language and Statistics II, 2007.

J. C. Brown, G. A. Frishkoff, and M. Eskenazi, "Automatic question generation for vocabulary assessment", in Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2005, pp. 819–826.

S. Bird, E. Klein, and E. Loper, Natural language processing with Python. O'Reilly Media, Inc., 2009.

R. Brachman and H. Levesque, Knowledge representation and reasoning. Elsevier, 2004.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge", in Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, 2008, pp. 1247–1250.