A Transfer Learning Approach Interaction in an Academic Consortium

Popescu Doru-Anastasiu[1](✉), Cristea Daniela-Maria[2], and Bold Nicolae[1,2]

[1] Department of Mathematics and Computer Science, Pitești University Center, National University of Science and Technology POLITEHNICA Bucharest, Bucharest, Romania
dopopan@gmail.com

[2] "1 Decembrie 1918" University of Alba Iulia, Alba-Iulia, Romania
daniela.cristea@uab.ro

Abstract. This paper focuses on presenting a methodology for designing an assessment test that takes into account the degree of difficulty of the items and the overall test structure in an integrated academic environment. This integrated approach allows for the collection of assessment items at an academic consortium level, making them applicable to similar faculties within a group of institutions. The proposed model consists of two main components: the item collector and the test generator. The item collector gathers assessment items from various academic institutions within the consortium. These items are then utilized by the test generator, which automatically generates tests using a combination of evolutionary algorithms (genetic-based) and machine learning (ML) techniques. The genetic-based algorithm and ML methods play a key role in determining the composition and structure of the assessment tests.

Keywords: Assessment consortium · Genetic algorithm · Transfer Learning

1 Introduction

This paper is addressing a research problem about automated test generation and presenting a methodology for designing an assessment test that takes into account the degree of difficulty of the items and the overall test structure in an integrated academic environment. In universities, there are many joint courses, especially in university consortia. The degree of preparation of students can be best determined by tests with the same items. In order to generate assessment tests that can be used in several universities in the consortium, we propose a model web platform containing two components: Assessment - Item Collection (A-IC) and Assessment - Test Generation (A-TG) using ML-based tools. The novelty brought by the paper is related to the usage of an integrated assessment tool within an academic consortium, formed by several academic institutions. This characteristic can help with a standardization of the assessment within the academic environment.

Suppose we want to monitor all the assessments inside a consortium and only want to process the questions that relate to courses. There are lots and lots of assessments that are being written each day. We wish to build an automated system (ML classifier) that would filter out all the irrelevant ones, and only forward the appropriate ones to certain courses. In general, when faced with such challenges, we need to either **get training** data or find datasets that we can use for training, and then apply (transfer) the models learned on somewhat different data types. This is sometimes referred to as **transfer learning** although, this phrase also denotes instances where we take a trained model and retrain it on a new kind of data.

The classifier serves a threefold purpose: (1) it employs machine learning techniques to generate keywords for each user-provided question; (2) it leverages machine learning to create question-and-answer groups; and (3) it utilizes genetic algorithms to generate tests.

In this matter, the paper presents several sections regarding the topic. The Literature review Sect. 2 presents state-of-the-art research regarding educational assessment and the usage of genetic algorithms and machine-learning-based tools used for educational purposes. The Model description Sect. 3 presents the description of the model, its components and functionalities, and Results and discussions Sect. 5 related to the implementation of the model for the genetic algorithm test generation with respect to the user requirements. A Conclusions Sect. 6 summarizes the entire research conducted for the paper.

2 Literature

The research in the automated educational assessment area is mainly conducted in several directions, the main two of them being Question Generation (QG) and Answer Evaluation (AE), according to [4]. While the QG research is mainly conducted on the automated generation of questions from a corpus of text using specific language processing methods, the AE research is focused on the analysis of the assessment based on the responses given to the item used for assessment.

Regarding the generation of assessment tests in the context of the existence of several requirements, special attention can be given to the usage of genetic algorithms (GA) with optimal solutions for statistical issues [17], management [5,10] or healthcare [20].

The research assesses competencies needed for university, guiding the development of the RespectNET training program. The matrix of competencies, as outlined in the authors [18] empirical study is categorized into four areas. For each competency, a set of descriptors has been provided, drawing from the European Reference Framework of Competences, the OpenEdu Framework, and the European Framework for the Digital Competence of Educators (DigCompEdu). Additionally, descriptors were collaboratively established by the RespectNET partners consortium. This comprehensive matrix of competence descriptions serves as a foundational resource for the development in designing course modules.

In [9], the author presents a Metaverse education framework in higher education, focusing on a consortium university in Korea. This university has created a shared learning environment through a consortium involving local governments, universities, and companies. They have opened a Metaverse virtual campus to bridge geographical gaps and utilize an LMS system for learning management. Despite differences between the website and Metaverse, the paper outlines suggestions for building a new learning system by leveraging their respective advantages.

In [8], three quality tools were utilized as ad hoc Quality Management (QM) measures for protocol generation/harmonization and data/knowledge exchange, including Sharepoint, Sample Tracker, a self-developed tool for progress monitoring, and Audits, an online auditing process to oversee and assess network activities.

A survey [22] gathered data from around 1,500 students about their academic experiences in fall 2020. It included questions about academic experiences, as well as subsets of questions related to specific topics and modalities of enrollment. The approach of assigning students to answer questions based on their enrollment modality including ensuring relevant feedback, increasing survey completion rates, and allowing variations in questions based on modality types.

The authors [15] discuss a type of optimization algorithm that is inspired by genetics, using notions such as chromosomes, genes, mutation, and crossover. The fitness of a sequence is determined by the number of keywords it shares with the user-defined set of keywords. The chromosomes are ordered by their fitness and the input data is made up of various parameters including the number of tests, the number of generations, and the user-defined keywords. The output data includes the first k solutions and the number of matching keywords for each sequence. In [2], some systems use distributional similarity techniques or extract words from text content for multiple options generation. Other frameworks use deep reinforcement learning for automatic question generation from corpora. The types of generated questions can be categorized by Bloom's taxonomy. Mostow and Chen [13] developed an automated reading tutor that uses automatic question generation to improve students comprehension of text, while only 35.6% of generated questions were deemed acceptable, the accuracy of detecting counterfactual questions was high at 90.0%. Mitkov and colleagues [12] developed a system to generate multiple-choice closed questions with natural languages processing techniques For 1000 question items, the development cost would require 30 h of human work using the system, while 115 h would be required without using the system. In [14], there were used learning materials from a graduate-level introductory data science course at an R1 university in the northeastern United States. The course has six conceptual units and sixteen modules, each consisting of several data science topics. Students also complete seven hands-on coding projects, which are evaluated by an automatic grading system. The authors focused on generating questions from the textual content of the six units in the course using a pipeline. Robinson [19] distinguishes task complexity (cognitive factors), task difficulty (learner factors), and task conditions (interaction factors), meanwhile Campbell [3] contrasted multiple views of complexity, such as a psychological experience, a task-person interaction, and a function of objective characteristics, and created a typology of complex tasks.

In the context of language tasks, Mesmer et al. [11] explicitly distinguish text difficulty (based on the performance of readers) and text complexity (based on textual elements). Beckmann and Goode [1] distinguish apply this distinction to the domain of complex problem-solving, such as controlling a dynamic system with feedback loops.

Gkatzia and Mahamood [6] conducted a study analyzing a corpus of 79 conference and journal papers on NLG published between 2005–2014, which revealed the increasing use of automatic evaluation over human evaluation and intrinsic evaluation over extrinsic evaluation.

3 Model Description

The main variables of the model are the questions, items named after the generation process, the test, the requirements, and the generation mechanism. A question is a particular case of an item, which can consist of a task or an exercise, which is why questions will be considered particular cases of items, and we will refer to questions, tasks, exercises, etc. as items. A set of items with specific properties forms an assessment test (Fig. 1).

The paper focuses on the improvement of the initial model with the automated generation of keywords for each item, which is crucial for the requirement related to the choice of the question based on the subject.

In addition to the three basic components of the model, we also consider the set of constraints (ConSet) formulated by the user, consisting of a list of parameters that can be given various values and based on which these tests are generated. The list of parameters studied so far in previous work includes:

- the subject/subjects that want to be treated in the evaluation;
- the degree of difficulty of the assessment sequence;
- the theoretical/practical report of the evaluation sequence;
- the predominant type of items within the evaluation sequence;
- the maximum time to solve the sequence of items.

Regarding the restriction on the subject of an educational assessment test from the perspective of the subject, there is the problem of determining the main subject of an item, in such a way that it is selected within the assessment test that wants to deal with that subject. In this sense, the determined solution is to describe the item through keywords that best describe that item. Going further with the implementation of the solution, the selection of questions with a certain topic can be done using the user setting the respective topic by a keyword that will filter those items that are described by the chosen keyword. Thus, the problem of describing each item by a series of keywords is raised. This description can be done through the establishment by a human user of these

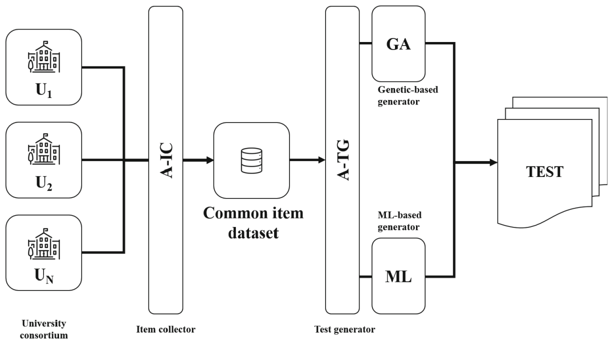


Fig. 1. Visual representation of the model

keywords, but this method is not practical. Therefore, another way of generating these keywords, as well as the process of dynamically extracting some items from a body of text, includes the use of natural language processing (NLP).

To accomplish this, will build a really quick classification model with a simple data pipeline to process the text data, using the Scikit-learn library. Initially, we employ stop-words to remove frequently occurring, non-informative words from our vocabulary. Next, we utilize the term frequency-inverse document frequency (TF-IDF) method to transform the articles into numerical vectors, preserving the importance of terms within the corpus.

The final vectors obtained from the TF-IDF transformation are used to train a Bayesian classifier. The Bayesian classifier is chosen as the first NLP model due to its simplicity and effectiveness in handling text data. The classifier uses probabilistic principles to make predictions based on the features derived from the TF-IDF vectors. To further optimize the performance of the model, we employ grid search to explore different hyperparameter combinations and identify the best configuration for the classifier.

3.1 TF-IDF Digression

The inverse statement frequency is a measure of how much information the word provides, i.e., if it's common or rare across all statements. TF-IDF takes all pieces of text, does a word frequency statistic on them, and then normalizes those frequencies by how common that word is (in general). The TF-IDF (term frequency-inverse document frequency) is defined in [7] as follows:

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D) \tag{1}$$

- $\mathrm{tfidf}(t, d, D)$ represents the TF-IDF score of a term t in a statement d within the statement set D.
- $\mathrm{tf}(t, d)$ denotes the term frequency of a term t in a statement d, which measures how frequently the term appears in the statement.

- $\mathrm{idf}(t, D)$ stands for the inverse statement frequency of a term t within the statement set D, which quantifies the rarity of the term across the entire set of statements.

In the case of the term frequency TF(t, d), the simplest choice is to use the raw count of a term in a statement, i.e., the number of times that term t occurs in statement d.

3.2 TF-IDF Transformer

In our investigation of the TfidfVectorizer, we performed a TF-IDF transformation on the entire training data, computing the inverse document frequency (IDF) on all statements, which resulted in an internal state of IDF values. Each statement is represented as a vector in a high-dimensional space, where each feature corresponds to a unique word in the corpus.

Later, when we applied the tf.transform() method to an individual statement, it computed the term frequencies (TF) for that specific statement and multiplied them with the precomputed IDF values from the internal state. This yielded the TF-IDF scores for all words in the vocabulary, corresponding to the given statement.

When we examined the result of this transformation, we encountered a sparse matrix representation (1, 3113) for a single statement, signifying 3113 TF-IDF features or values, one for each word in the vocabulary. As expected, the vast majority of these values were 0, as each statement typically contains only a few of the possible words.

4 Algorithms

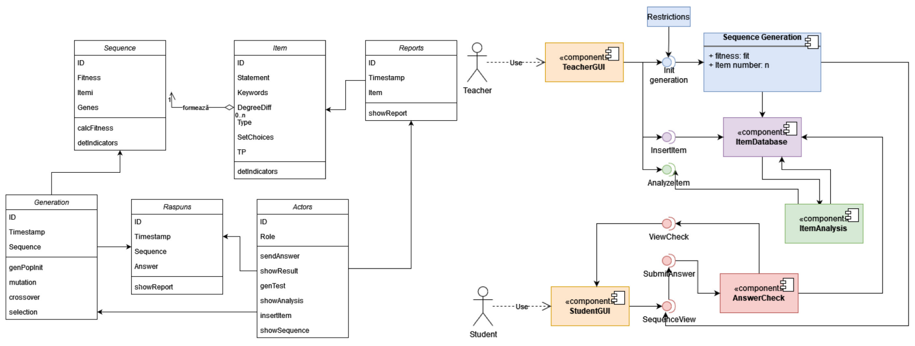
4.1 Item Collection (A-IC)

A-IC Components. The item collection (A-IC) component is the one responsible for the creation of a dataset of items and the continuous improvement and update of the items present in the dataset. The model is based on the existence of the item and its characteristics. In this matter, the item, which will be denoted further by q $(id_q;\ st_q;\ dd_q;\ v_q;\ t_q,\ ans_q)$ is an object formed of the next components:

- the identification part of the item id_q, which consists of an identification particle of the item; in our model, id_q is considered to be a number and will play an important role in the implementation phase ($id_q \geq 1$, $id_q \in \mathrm{N}$);
- the statement st_q, formed of the phrase or a set of phrases that contain the requests of the item (either questions, statements, or exercises); the statement is an important part regarding the automated generation of tests;
- the set of keywords kw_q, of cardinal nr_{kw}, which contains the list of keywords that describe the item, either its category or the main topic of the item. A keyword kw_{q_i}, $i = 1$, nr_{kw}, can be established by a human operator, or it can be generated using specific methods of language programming, such as Natural Language Programming (NLP);

– the degree of difficulty dd_q, $dd_q \in [0, 1]$; which is firstly pre-established using the method presented in [16]. The degree of difficulty is then calculated as the ratio between the number of incorrect answers at the item and the total number of answers, being continuously updated by the answers given to the item;
– the item type t_q, $t_q \in \{\text{'m','s','e'}\}$, which shows the type of the item, whether it is multiple-choice based (m) or the answer is a textual one, given by the user, in case of short (s) and essay (e) types.
– choices set v_q (when $t_q =$"m"), which can be formed of a list of two or more possible answers when the item type is multiple or is null when the item type is not ($v_q \geq 1$, $v_q \in \mathrm{N}$);
– the correct answer of the item ans_q, which contains the correct answer of the item; it has the form of:
  - a choice identifier, such as a letter (a, b, c, ...) or a number (1, 2, 3, ...), in case of multiple-choice items ($t_q =$ "m");
  - a real number, in case of numerical answers;
  - a text, in case of short or essay items ($t_q \in \{\text{'s', 'e'}\}$).

The component diagram, shown in Fig. 2b, emphasizes the modular character of the program system, starting from the three main components of the general model. The diagram presents aspects related to restrictions, interfaces, databases, and actors involved in the system (Teacher and Student). The component diagram helps to visualize the behavior of the system in the context of a real-world application. Also, the Item class shows the modality of the storage of the generated items in the database, as shown in Fig. 2a.



(a) The Item class and its relation with the other classes (b) Arhitecture model, comprising the item generation and storage

Fig. 2. Model arhitecture

4.2 Test Generation (A-TG)

A-TG Components. An important component of the test generation model is a test T (S, DD), which is considered a set of items q_i, $i = 1, \|S\|$, where S is the set of items that form the test, dd_i is the degree of difficulty of the item i and DD is the degree of difficulty of the test:

$$DD = \sum_{i=1}^{S} q_{dd_i} \tag{2}$$

A-TG Functionality

A-TG Using Genetic Algorithms. The pseudocode of the genetic algorithm (GA) is presented in Algorithm 1.

Algorithm 1 Genetic Algorithm for A-TG

```
1: Input: $population\_size, generations$
2: Output: $\mathbf{X*}(itemIDsequence)$
3: Population initialisation $X_i(i = 1, 2...n)$
4: Calculate the fitness value of every chromosome
5: Selection
6: $\mathbf{X*} \longleftarrow$ the best chromosome
7: $t \longleftarrow 0$
8: while ($t < generations$) do
9:     for (every chromosome) do
10:        $\mathbf{X}(t+1) \longleftarrow Mutation(\mathbf{X}(t), \mathbf{Y}(t), \mathbf{X*})$
11:        $\mathbf{X}(t+1) \longleftarrow Crossover(\mathbf{X}(t), \mathbf{X*})$
12:    end for
13:    Calculate the fitness of the modified chromosome;
14:    $Selection$
15:    $\mathbf{X*} \longleftarrow$ the best chromosome with the minimum fitness
16:    $t \longleftarrow t + 1$
17: end while
18: return $\mathbf{X*}$
```

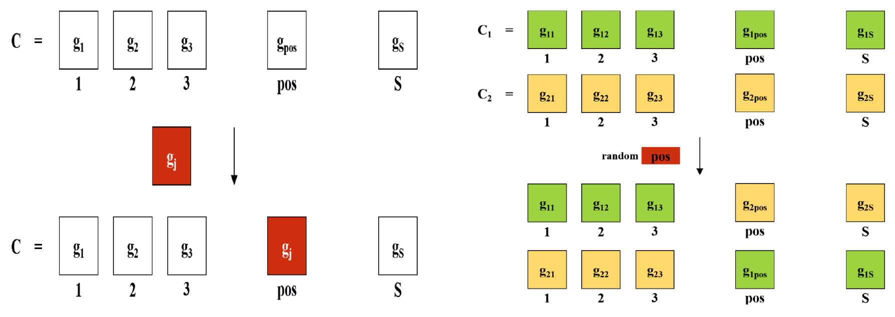
The genetic algorithm codifies items as genes within a chromosome, which will represent a test. In this matter, the next components will be needed:

- a chromosome C, which codifies a test, formed of the next components:
  - order number id, $id \in \{0, \ldots, NrI\}$, where NrI is the total number of items within the database;
  - the gene set $G_j = \{g_j | j \in \{1, \ldots, S\}\}$, where $G = S$; $j = 1, nr_T$, $nr_T = n_1 + n_2 + ... + n_i$, $i = 1, n$;
  - the fitness function f, which calculates the closeness of the degree of difficulty of the chromosome C to the desired degree of difficulty DD. The fitness function calculation also gives the flexibility of choosing a lower or a higher level of difficulty, a higher value of min meaning a higher level of difficulty, and a lower value of min combined with a lower

value of *max* meaning a lower level of difficulty. The fitness function is defined as follows:

$$f(C) = \left| \sum_{i=1}^{S} q_{dd_i} - \frac{min + max}{2} \right| \tag{3}$$

– the genetic operators, the ones used in the presented model being:
  - the generation of a chromosome $GenC$, an operator that forms a new chromosome;
  - the mutation of a chromosome $MutC$, which can be defined as follows: for a given chromosome C, a random position within the chromosome pos, $pos = 1, S$, and a randomly selected gene g_j, $j = 1, NrI$, the mutation operation is defined as the shift of the gene g_{pos} with the gene g_j.
  - the crossover of a chromosome $CsvC$, which can be defined as follows: for two given chromosomes C_1 and C_2 and a randomly selected position within the two chromosomes pos, the first part of the chromosome C_1 up to the gene g_{pos} is combined with the second part of the chromosome C_2 from the gene g_{pos} to the end of C_2 and the first part of the chromosome C_2 up to the gene g_{pos} is combined with the second part of the chromosome C_1 from the gene g_{pos} to the end of C_1, resulting in two new chromosomes C_1' and C_2'.



(a) The $MutC$ operator (b) The $CsvC$ operator

Fig. 3. Visual representation of the operators

A-TG Using ML-Based Tools - Measurement Framework Architecture. In order to compare the metrics of multiple models consistently, we have established a measurement framework to allow us to assess their performance. The core component is a measure (model, x, y) function that takes the model, accepts a list of strings as input (x), and returns a list of numbers representing the predictions with the correct labels (y) adhering to a classification report regarding the metrics.

Model 1. As part of the implementation process, we have created our own model 1 function that serves as a wrapper for the pipeline model. The prediction function takes a list of input texts (x), tokenizes it into words, checks if any of the words intersect with the predefined parameter sets and returns a list of predictions associated with these texts. In our case, the predictions are the classifications resulting from the classifier model on the test dataset (classification_test_inputs).

Model 2. Nevertheless, some optimisations are required to find the best overall model based on the prediction classifications resulting from the GridSearchCV model.

Model 3. The previous model used all the features (words) in the vocabulary, but realistically when dealing with lots and lots of files, the vocabulary will increase to hundreds of thousands of words and thus each TF-IDF entry will be a vector of size (1, x00 000), ridiculously large. This is sometimes called "the dimensionality curse" and it happens when on small scale things seem to be working but in a production environment, the size of the data starts to make the approach intractable. In order to solve this, we will be using an improvement over the previous model, namely we will not keep the full vocabulary while computing the results, but only some of the most relevant words getting the SelectKBest instance from the pipeline where all the words that the feature selection step has chosen and printed all the selected words, in descending order of their score (the most important ones are at the top).

Model 4. If we are looking closely, there are many words that have the same meaning but are slightly different (because of the context in which they are used). In an example resulting from the dataset, we have "thanks" and "thanksgiving", so we would like to normalize them as much as possible to keep the overall vocabulary small. This process can be accomplished through two approaches, *stemming* and *lemmatization*. So far, we implemented our own transformer and do both at the same time. NLTK already provides us with convenient objects that we can use, where we can inject this class into the TfidfVectorizer as a custom tokenizer function and end up a mixed object where we won't be able to use the stop_words = 'english' parameter because it will be applied to that list as well. Considering this, we got a standalone StopWordsTransformer that will filter out the stop-words and a standalone LemmaStemmerTransformer that should take the non-stop words and do lemmatization and semisation on them. These two custom transformers should be placed in front of the TfidfVectorizer.

The initial model performances yielded F1-scores of **0.79** for the first model and **0.74** for the second one. Upon invoking the measure function with the test data (classification_test_inputs) and correct labels (classification_test_labels), all three models achieved perfect classification for each class, resulting in a harmonic mean **F1-score** of **1**. The dataset is well-balanced, as it contains a similar number of instances for each class (Fig. 3).

5 Results and Discussions

Firstly, we will showcase results for items with manually set keywords, as presented in Table 1. Secondly, we will present results for items with automatically set keywords, as shown in Table 2. The shown tables present similar results, with slightly finer results for the automated generated keywords, given by the user refinement. In order to show the effectiveness of the NLP automated extraction of the keywords, we have used the WordNet library to automatically extract two keywords from each statement of the questions. The compared versions of the keywords are shown in Table 3.

A great deal of importance must be brought to refining the generation of keywords, with the additional feature of generating synonyms to the found keywords. Also, it can be observed that there are significant differences between the user keywords and the generated keywords, due to the close relationship between the vocabulary of the statement and the generated keywords. This aspect can be improved by refining the algorithm with the addition of finding synonyms or words close to the meaning of the ones in the statement.

Essentially for optimizing the model accuracy, we explored some techniques to work effectively with numerical data and algorithms to enhance the performance of the model as presented in Table 4, which has been transformed into a numerical vector format.

Feature Selection. Based on the experiments, we employed the SelectKBest and SelectPercentile methods to identify the most informative features from our dataset, where was achieved an accuracy of **0.53**.

Ensemble Methods. To further enhance the accuracy, we leveraged the power of ensemble methods such as Random Forest and Gradient Boosting, where was obtained an improved accuracy of **0.6**.

Hyperparameter Tuning. By using GridSearchCV, we fine-tuned the hyperparameters of our model to identify the optimal combination, resulting in an accuracy of **0.67** and demonstrating the importance of selecting appropriate hyperparameters for optimal model behavior.

Bayesian Classifier. We implemented a Bayesian classifier on top of the final vectors obtained from the TfidfTransformer and SelectKBest. The MultinomialNB classifier was utilized and got a comprehensive classification report with an overall score of **0.57**.

K-Means Clustering. For a deeper understanding of the data, we applied K-Means clustering on the TF-IDF matrix of the text data. This technique allowed us to assign data points to different clusters, where the best estimator achieved a score of **0.67**, indicating the effectiveness of K-Means clustering in grouping similar data points together.

Figure 4a shows that the best values for the closeness of the degree of difficulty can be found for a larger initial population size. In this matter, minimum and maximum values are the lowest for an initial population size of 50 from the ones taken into consideration.

Figure 4b shows that the best values for the closeness of the degree of difficulty can be found for a lower number of generations. In this matter, minimum and maximum values are the lowest for a number of 200 generations from the ones taken into consideration.

Table 1. Obtained values of the test for the input data for the manual setting of the keywords

ID	Statement	University	Diff
57	What is the result of the expression: (100 = 101) AND (2*3 < 6)	UPIT	0.54
65	Which function in Excel returns the sum of a range of numbers?	UPIT	0.34
21	What given number is prime?	UPIT	0.34
8	Which of these is not an item of hardware to do with computers?	UAI	0.99
42	What is wrong with the following piece of code: includ ?	UAI	0.89
55	What is the result of the expression: (5 > 7) OR (0 < 2 * 5 < 15)	UPIT	0.87
45	What is the name of the direction of a page used for viewing and printing?	UPIT	0.17
17	What is a URL?	UPIT	0.38
13	Which is the best application to use to write a letter?	UPIT	0.16

Table 2. Obtained values of the test for the input data

ID	Statement	University	Diff
6	The Microsoft Word program is	UAI	0.34
10	What is the name of the direction of a page used for	UPIT	0.35
7	The extension of a file created in Word is:	UAI	0.82
9	The process of removing an unwanted part of an image	UAI	0.99
12	What is the term for unsolicited emails?	UPIT	0.59
66	What is a TCP/IP?	UPIT	0.95
14	URL means:	UPIT	0.16
23	The process of arranging the elements of a column...:	UPIT	0.65
23	Which function in Excel returns the average...	UPIT...	0.16

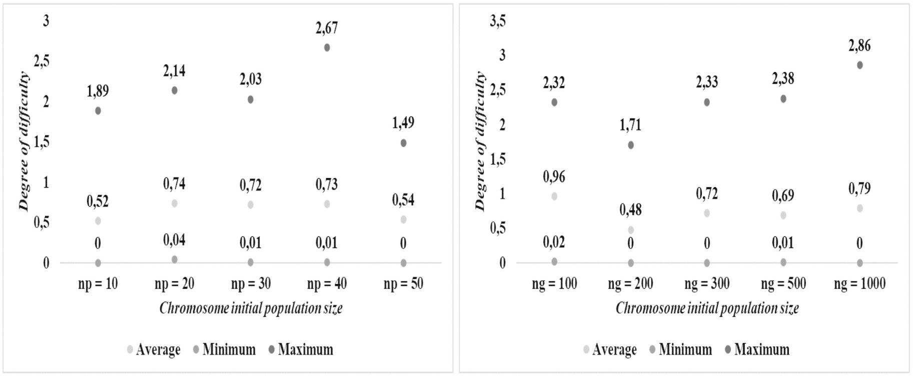
Table 3. Example of automated generated keywords for random 10 questions in the database

Item ID	User keywords	Generated keywords
1	operating, system	operating, following, system
2	operating, system	operating, mobile, system
3	operating, Windows	systems, differences, windows, linux
4	path, file	path, desktop, file, hello.txt
5	operating, Linux	operating, words, system, linux
6	C++, library	library, files
7	string, palindrome	code, string, palindrome
8	Android, code	code, wrong, android
9	encryption, data	data, confidential
10	hardware, item	item, hardware, computers

Table 4. Model Performance Experiments

Experiment	Accuracy	Notes
Feature Selection	0.53	Statistical relationship between features and target variable considered
Ensemble Methods	0.6	Random Forest and Gradient Boosting used for model combination
Hyperparameter Tuning	0.67	GridSearchCV and RandomizedSearchCV to find optimal hyperparameters
Bayesian Classifier	0.57	MultinomialNB classifier on TfidfTransformer and SelectKBest vectors
K-Means Clustering	0.67	K-Means applied on TF-IDF matrix of text data

In real-world scenarios, datasets [21] often exhibit class imbalance, even if are well-balanced, with a similar number of instances for each class. In this case, the F1-score metric does not accurately reflect the model performance. The inefficiency of traditional models in handling enormous TF-IDF vectors is emphasized and results demonstrate the importance of careful feature selection, ensemble methods, hyperparameter tuning, and clustering techniques to achieve higher accuracy and performance in our model. These findings are crucial for enhancing the overall quality and reliability of our model, making it more suitable for real-world applications.



(a) The variation based on the initial population size (b) The variation based on the number of generation NG

Fig. 4. The variation of the degree of difficulty

6 Conclusions

In this work we conducted an experiments to evaluate a method for assessment test generation in an academic consortium. The proposed model consists of two components. The first component, namely, A-IC, aims to collect items for the assessment and creating the item database. The second component, namely, A-TG, aims at generating the test using genetic algorithms and establish a measurement framework to compare multiple models. In particular, Bayesian classifier is employed to classify the item based on the TF-IDF representation.

A great deal of importance must be brought to refining the generation of keywords, with the additional feature of generating synonyms to the found keywords. Also, it can be observed that there are significant differences between the user keywords and the generated keywords, due to the close relationship between the vocabulary of the statement and the generated keywords. This aspect can be improved by refining the algorithm with the addition of finding synonyms or words close to the meaning of the ones in the statement.

Acknowledgment. Cristea Daniela Maria acknowledges the support from the Romanian Ministry of Education, under contract no. 3721/2023, and of the University "1 Decembrie 1918" of Alba Iulia https://en.uab.ro/centre/1-center-for-scientific-research/

References

1. Beckmann, J., Birney, D., Goode, N.: Beyond psychometrics: the difference between difficult problem solving and complex problem solving. Front. Psychol. **8**, 1739 (2017). https://doi.org/10.3389/fpsyg.2017.01739
2. Blšták, M., Rozinajova, V.: Automatic question generation based on sentence structure analysis using machine learning approach. Nat. Lang. Eng. **28**, 1–31 (2021). https://doi.org/10.1017/S1351324921000139
3. Campbell, D.: Task complexity: a review and analysis. Acad. Manag. Rev. **13**, 40–52 (1988). https://doi.org/10.5465/AMR.1988.4306775
4. Das, B., Majumder, M., Phadikar, S., Sk, A.A.: Automatic question generation and answer assessment: a survey. Res. Pract. Technol. Enhanc. Learn. **16**, 1–5 (2021). https://doi.org/10.1186/s41039-021-00151-1
5. Davis, L.: Job shop scheduling with genetic algorithms. In: Proceedings of the First International Conference on Genetic Algorithms and their Applications, pp. 136–140. Psychology Press (2014)
6. Gkatzia, D., Mahamood, S.: A snapshot of NLG evaluation practices 2005–2014 (2015). https://doi.org/10.18653/v1/W15-4708
7. Hamdaoui, Y.: TF(term frequency)-IDF(inverse document frequency) from scratch in python. Towards Data Sci. **23**, 2022 (2019). https://towardsdatascience.com
8. Hülsemann, M., et al.: Introducing quality measures in an academic research consortium: lessons and recommendation from implementing an ad hoc quality management system for organ model research. EMBO Rep. **23**, e55095 (2022). https://doi.org/10.15252/embr.202255095
9. Jeong, Y., Choi, S., Ryu, J.: Work-in-progress-design of LMS for the shared campus in metaverse learning environment. In: 2022 8th International Conference of the Immersive Learning Research Network (iLRN), pp. 1–3 (2022). https://doi.org/10.23919/iLRN55037.2022.9815909
10. Lee, C.K.H.: A review of applications of genetic algorithms in operations management. Eng. Appl. Artif. Intell. **76**, 1–12 (2018)
11. Mesmer, H., Cunningham, J., Hiebert, E.: Toward a theoretical model of text complexity for the early grades: learning from the past, anticipating the future. Read. Res. Q. **47**, 235–258 (2012). https://doi.org/10.1002/rrq.019
12. Mitkov, R., Ha, L., VARGA, A., Rello, L.: Semantic similarity of distractors in multiple-choice tests (2009). https://doi.org/10.3115/1705415.1705422
13. Mostow, J., Chen, W.: Generating instruction automatically for the reading strategy of self-questioning (01 2009). https://doi.org/10.3233/978-1-60750-028-5-465
14. Nguyen, H.A., Bhat, S., Moore, S., Bier, N., Stamper, J.: Towards generalized methods for automatic question generation in educational domains (2022). https://doi.org/10.1007/978-3-031-16290-920
15. Popescu, A.D., Bold, N., Nijloveanu, D.: A method based on genetic algorithms for generating assessment tests used for learning. Polibits **54**, 53–60 (2016)
16. Popescu, D.A., Bold, N.: The development of a web application for assessment by tests generated using genetic-based algorithms. In: CEUR Workshop Proceedings (2016)
17. Quirós, P., Lasheras, F.S.: Methodology for the projection of population pyramids based on monte carlo simulation and genetic algorithms. Appl. Intelligence pp. 1–18 (2023)
18. Robescu, D., Reiner, S., Trunk, A.: Toward a matrix of competences for respectful communication in the university-civil society context (2023). https://doi.org/10.21125/edulearn.2023.0217

19. Robinson, P.: Task complexity task difficulty and task production: exploring interactions in a componential framework. Appl. Linguist. **22**, 27–57 (2001). https://doi.org/10.1093/applin/22.1.27
20. Sharma, S., Kumar, V.: Application of genetic algorithms in healthcare: a review. Next Generation Healthcare Informatics, pp. 75–86 (2022)
21. Tiwari, R.: Imbalanced learning in fraud prevention: Views and solutions from the trenches. CueNex, February 2023. https://medium.com/cuenex, February 2, 2023
22. Whelehan, D.: Students as partners: a model to promote student engagement in post-COVID-19 teaching and learning. J. Higher Educ. (2020)