Francisco Jose De Caldas District University
Faculty of Engineering

Steven Navarro Parrales
20221020048

Juan David Amaya Patiño
20221020057

David Santiago Garcia Galeano
20231020158

Eng. Carlos Andrés Sierra
SYSTEMS ANALYSIS AND DESIGN
Group: 020-82

Bogotá
2025

Competition link:
https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/overview

Competition Overview: Predicting the affinity between misconceptions and incorrect answers (distractors) in multiple-choice questions.

teaching-learning processes through artificial intelligence. One of the main findings is the efficient automation of misconceptions labeling, which addresses an important pedagogical need and how metrics such as MAP@25 and computational constraints not only guide the development of effective solutions, but also applicable in real-world contexts. Together, this system represents an effective synergy between technology, education, and data science to address a high-impact educational problem.

Simulation Report

1. **Introduction.**
   This Simulation Report evaluates a system designed to detect and classify student misconceptions in mathematics using AI and natural language processing. Based on the design proposed in Workshop 2, the system analyzes distractors from multiple-choice questions and links them to common misconceptions using semantic similarity and machine learning classification.
   The core dataset, from the Kaggle competition "Eedi – Mining Misconceptions in Mathematics", contains 2,586 labeled misconceptions. To assess the system's performance under different levels of complexity, three reduced datasets of 100, 500, and 1000 misconceptions were used in simulation.
   The simulation tests the system's scalability, accuracy, and stability, using structured steps such as pair generation, affinity scoring, filtering, and classification. Key metrics include affinity range and classifier accuracy. This report demonstrates the system's ability to maintain high performance while adapting to increasing data complexity, confirming its potential for educational use.

2. **Scenario of the Simulation.**
   a. **Datasets.**
      The simulation was built upon the dataset provided by the Kaggle competition **"Eedi – Mining Misconceptions in Mathematics"**, which focuses on predicting the semantic affinity between student misconceptions and distractor choices in multiple-choice mathematics questions. The dataset provided was *misconception_mapping.csv* in this .cvs have a list of 2586 misconception, these have a format:
      - *MisconceptionId*: Unique numerical identifier that serves as a reference key for each misconception.
      - *MisconceptionName*: A text description detailing the misconception, often formulated in natural language that mirrors how a student might misunderstand a concept.

      These misconceptions cover a wide array of mathematical domains, such as geometry, algebra, percentages, and number theory. This semantic is crucial, as it enables the system to train the model capable of distinguishing nuanced conceptual errors.

3. Methodology.

a. Model Test.

For the test of the system model and to see the how sensitive dependence is, the potentials emergent behaviours we took the decision to reduce the original dataset *misconception_mapping.csv* and make three .csv to do the test:

- *misconception_mappingV1.csv:* it has 100 misconceptions, with their ID and Name.
- *misconception_mappingV2.csv:* it has 500 misconceptions with the ones from *V1*, with their ID and Name.
- *misconception_mappingV3.csv:* it has 1000 misconceptions with the ones from *V2*, with their ID and Name.

This reduction allows the simulation to evaluate changes in classifier performance, affinity stability, and error propagation.
Additionally, the distractors provided by the Kaggle competition use consistent formatting and semantic diversity to stress-test the model and classifier logic under varied conditions.

b. Steps for the simulation.

i. Load Dataset:
   1. A .csv file containing 100/500/1000 misconceptions is loaded.
   2. It will be to proceed with eighth simulations for each .csv file.

ii. Generate Pairs:
   1. The system will generate 100 questions–distractors pairs for analysis.

iii. Affinity Calculation:
   1. Measures semantic similarity or relatedness between distractors and misconceptions.
   2. Affinity threshold ≥ 0.15 is applied to retain relevant pairs (filtered subset).
   3. This filtering stage echoes the Filter Affinity and Less Similar components of the architecture described in Workshop 2, ensuring that only high-quality associations are passed on.

iv. Classifier Training:
   1. A machine learning classifier is trained to label the filtered pairs with specific misconception categories.
   2. Performance metric: Classifier accuracy, indicating strong performance in matching distractors to the correct misconception category.

v. Analysis Report for the Simulations:

1. Outputs key metrics (affinity stats, category distributions, predictions).
2. Classifier accuracy.
3. Lists most frequent misconceptions and their examples.

This testing methodology directly reflects the architectural principles outlined in Workshop 2, where sensitivity and chaos are managed via modularity, semantic filtering, and iterative verification. Through this controlled setup, the simulation could reveal how changes in input size and diversity affect affinity thresholds, classifier decisions, and final system output.

This methodology reflects a systematic, iterative, and modular approach to simulation. It allows for real-world modeling of how educational AI systems behave under changing inputs, respecting principles from systems theory and chaos mitigation. The integration of randomness, threshold-based filtering, and rigorous classification makes the methodology not just technically sound but also educationally meaningful.

4. **Code highlights.**

   a. **Languages codes:**
   The simulation was implemented in Python using the **MisconceptionSimulator** class, which structures the entire process into an educational pipeline. The system uses object-oriented programming to manage tasks such as loading data from CSV, automatically generating question-distractor pairs, calculating semantic affinity with **SentenceTransformers**, performing relevance filtering, and training a classifier (**RandomForest**) to predict misconceptions. Tools such as **scikit-learn** and **pandas** are used, and good software practices are followed, including error handling and fallbacks if data is unavailable.

   b. **Prototype code:**
   The current code serves as a well-documented prototype of a microservices architecture, where each method represents a standalone service: **load_misconceptions()** serves as a data ingestion service with automatic validation and fallback, **calculate_affinity_score()** serves as a processing service combining semantic (70%) and lexical (30%) similarity to assess distractor quality, and **train_misconception_classifier()** serves as a ML inference service with data validation, adaptive train-test splitting, and evaluation metrics. Each method includes detailed comments explaining its purpose, parameters, return values, and implementation considerations, as well as robust handling of edge cases such as small datasets, insufficient classes for classification, and division by zero in similarity calculations.

c. **Chaos Theory:**
The simulation incorporates chaos theory principles through multiple random perturbation mechanisms and feedback loops: it uses stochastic selection in **generate_question_distractor_pairs()** where categories, questions, and distractors are randomly chosen to introduce controlled variability; it implements adaptive thresholds in **filter_high_affinity_pairs()** where small changes in the threshold can significantly affect the final dataset size; and it employs self-tuning parameters such as the train-test split size that dynamically adapts based on the available dataset size. The system is resilient, incorporating degradation mechanisms that allow it to continue functioning in the face of incomplete data, missing files, or boundary conditions. It maintains its core functionality and offers clear feedback through detailed logs and analytical reports.

5. Main Results.

a. Performance

With the simulations done, the tables will summarise the performance overall of the three versions used of the *misconception_mapping.csv* original dataset.

1. *misconception_mappingV1.csv*

Across all simulations, the most frequent categories in the associated misconceptions were:

Category	Frequency range
triangle_angles	14 to 23
percentage	14 to 21
algebra	6 to 15
square_root	5 to 11
geometry	0 to 3

Key Metrics Across All Simulations:

Metric	Range
High-affinity pairs	42 to 60 out of 100
Average affinity	Between 0.264 and 0.283
Max affinity	Constant at 0.431

Min affinity	Constant at 0.158
Classifier accuracy	From 0.875 to 1.000

2. *misconception_mappingV2.csv*

Across all simulations, the most frequent categories in the associated misconceptions were:

Category	**Frequency Range**
triangle_angles	16 – 23
percentage	8 – 25
square_root	8 – 15
algebra	4 – 11
geometry	1 – 4

Key Metrics Across All Simulations:

Metric	**Range**
High-affinity pairs	Between 48 and 64
Average affinity	Between 0.257 and 0.286
Max affinity	Mostly 0.431, some at 0.408
Min affinity	Between 0.158 and 0.167
Classifier accuracy	1.000 in all simulations

3. *misconception_mappingV3.csv*

Across all simulations, the most frequent categories in the associated misconceptions were:

Category	**Frequency Range**
triangle_angles	14 – 20
percentage	13 – 21
square_root	5 – 13

algebra	6 – 12
geometry	1 – 4

Key Metrics Across All Simulations:

Metric	Range
High-affinity pairs	Between 47 and 62 (out of 100)
Average affinity	From 0.261 to 0.278
Max affinity	Up to 0.431
Min affinity	Between 0.158 and 0.167
Classifier accuracy	0.824 – 1.000

6. **Findings.**

For the 1000(*V3*) Compared to the 100(*V1*) and 500(*V2*) versions, this set shows slightly more variation in classifier performance and affinity values, expected due to the dataset's increased complexity and diversity.

Across all dataset versions, the following misconceptions were the five most recurring errors:

1. Believes triangle angles don't sum to 180°.
2. Thinks percentage = just adding a "%" sign.
3. Believes the inverse of square rooting is doubling.
4. Believes non-unit multiples of vectors can be unit vectors.
5. Confuses area formulas of triangle vs. circle or other shapes.

For the accuracy range observer for each dataset version, with a small drop in the accuracy is expected with a more varied and noisy input, but accuracy still remains very high and above usable thresholds for educational applications.

Dataset Size	Accuracy Range
misconception_mappingV1.csv	0.875 – 1.000
misconception_mappingV2.csv	1.000 (perfect)
misconception_mappingV3.csv	0.824 – 1.000

For the affinity ranges with a minimum of 0.158 and maximum of 0.431 remained stable across all simulations done, and for the average affinity hovered around 0.26 to 0.28, regardless of dataset version.
Filtered pairs with affinity ≥ 0.15 ranged from 42 to 64 per simulation in all cases, consistently filtered about half the dataset, showing good balance between noise and relevance

7. Conclusions.

The workshop demonstrated the successful development and evaluation of a simulation pipeline designed to detect and classify student misconceptions using artificial intelligence. By progressively testing datasets of 100, 500, and 1000 misconceptions, the team validated the scalability, stability, and effectiveness of the system under increasing data complexity. The methodology involved loading misconceptions from .csv files, generating question–distractor pairs, calculating semantic affinities using sentence embeddings, filtering relevant pairs (affinity ≥ 0.15), and training a classifier—achieving high accuracy across all cases. Despite minor accuracy variations due to dataset size and diversity, the system maintained strong performance accuracy between 0.824 and 1.000 and a consistent affinity range average between 0.26 and 0.28, min 0.158, max 0.431. Across all simulations, the most frequent misconceptions were remarkably consistent, including misunderstanding triangle angle sums, misinterpreting percentages, confusing square root operations, and misapplying geometric formulas. The implementation in Python used well-structured, object-oriented code with microservices-inspired functions, integrating libraries such as scikit-learn and SentenceTransformers. The system also incorporated chaos theory principles, such as stochastic pair generation and adaptive thresholds, making it resilient to noise and variation. In conclusion, the workshop effectively combined systems analysis, educational theory, and machine learning to create a reliable and explainable model that supports automated misconception detection—highlighting its applicability for intelligent tutoring systems and real-world educational interventions.