AI In Education

Artificial Intelligence In The Education Ecosystem

MAJOR SPONSORS:

Vasile Rus, PhD

The University of Memphis

vrus@memphis.edu

Learning With Technology:
Data, AI, Technology Supporting All Learning & Instruction Environments

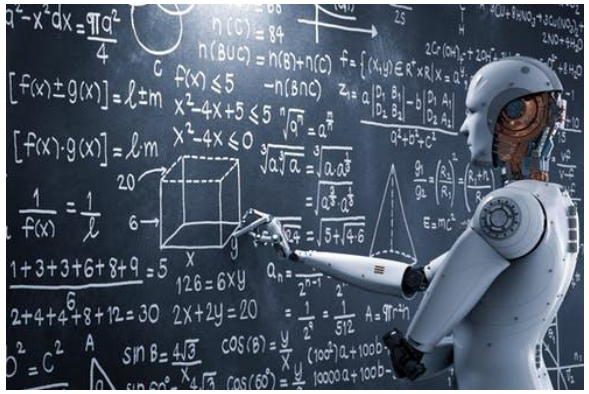


Technology just "observes and collects"





Blended-learning



Tech-driven/Human-behind-the-scenes

Level 5 self-driving/learning

Major Tutors

- **DeepTutor** (conceptual Newtonian Physics)
- MetaTutor – metacognitive support (biology/pre-med)
- Writing Pal – writing support
- DataWhys – data science tutoring
- **iCODE** – code comprehension

Outline

- Overview of the AIED landscape [*5 minutes*]
- Historical Perspective [5 minutes]
- Three Examples
  - iCODE: Adaptive Scaffolding of Learners' Code Comprehension Processes [*5 minutes*]
    - Funded by Department of Education [$2 million]
  - Neuro-Symbolic AI for Math Tutoring [*5 minutes*]
    - *Investigating techniques that couple Markov Logic and Deep Learning with applications to discovering strategies to improve STEM learning* – funded by NSF [$400k]
    - *ASTRA: AI-based Strategy Analysis* – funded by Gates Foundation [$500k]
  - AAR-Tutor: An After Action Review Tutor [*5 minutes*]
    - Funded by the Department of Defense [$1.4 million]
    - Part of the U.S. Army's Generalized Intelligent Framework for Tutoring (GIFT – gifttutoring.org)
- Discussion [*5 minutes*]

Overview of AIED

- Goal: using AI to understand, model, and scaffold students' learning process towards mastering a target domain
- Learning with technology: anytime, anywhere
- Learners: anyone, any age, any mastery level
  - Preschool
  - K-12
  - College
  - Adults
    - New recruits in the military
    - Prisoners/incarcerated people
    - Corporate personnel

Tutoring or One-on-One Instruction: One of The Most Effective Forms of Instruction

Intelligent Tutoring Systems

- Goal: Mimic the human tutor–learner interaction
- Dream: provide a tutor, human or computer-based, to every learner anytime/anywhere
  - Computer-based tutors could potentially scale up to all learners
- Fully adaptive instruction:
  - macro-level adaptation
  - micro-level adaptation
- Very different from other, widespread computer-based training (CBTs) systems
  - CBTs offer a one-size fits, mastery-learning based approach

Conversational Intelligent Tutoring Systems

- My focus is on **fully adaptive conversational Intelligent Tutoring Systems (cITSs)**
- All major challenges of dialogue systems
  - **Dialogue management**
  - **Natural language understanding, e.g., assessing the correctness of learners' natural language responses**
  - **Knowledge representation of a target domain**
  - **Natural Language Generation**
  - **Question Answering**
  - **Question Generation**
  - **Information Extraction**
- **Task-oriented approaches as a way to manage complexity**

Why Conversational AI Systems?

- **Natural**
  - Mimicking the human tutor – learner natural interaction
  - Socratic method, (one of) the most highly regarded instructional method
    - Goal in education: find the optimal sequence of questions to induce learning
- **Interactive**
  - Theoretical and empirical evidence that interactive collaborative learning is better (INTERACTIVE>CONSTRUCTIVE> ACTIVE > PASSIVE)
- **Encourage deeper, conceptual learning**
  - as students are required to explain their reasoning and reflect on their basic approach to solving a problem
- **Strong link between language and knowledge**
  - Natural-language based knowledge representations
- **Provide deeper insights into the learning process**
  - by revealing students' reasoning and mental model construction processes
  - enabling better detection of opportunities to scaffold students' learning by providing immediate corrective feedback, which is most effective.
- **Natural language interactions give students the opportunity to develop the language of their target professional communities** and therefore much-needed communication skills

A Long History …

- **TEACH**: A Program to Teach Programming (Robert et al., 1970)
- **CIRCSIM-Tutor** (Evens et al., 1997)
- **AutoTutor** (Graesser et al., 2005)

Quick Remarks

- TEACH:
  - Vague, overly optimistic Natural Language Processing (NLP)
- CIRCSIM-Tutor
  - Hard-core NLP (more symbolic NLP)
  - Scalability challenges
- AutoTutor
  - Loose input understanding based on Latent Semantic Analysis and regular expressions (more statistical NLP)
  - Simple generation and dialogue management based on a well-defined task structure (script)
  - More scalable

TEACH: A Program to Teach Programming

- “The TEACH system was developed at MIT to ease the cost and improve the results of elementary instruction in programming. To the student, *TEACH offers loosely guided experience with a conversational language which was designed with teaching in mind. Faculty involvement is minimal*.”
- The TEACH system was developed to provide ***virtually instructorless instruction*** in programming.

TEACH: Pedagogy

- Every important idea should be presented only after a need for it has been clearly established.
- ***Before an idea is presented to a student, he should be brought to the threshold of inventing the idea himself*.**
  - For example, he should have had to write a particular program segment several times within a single program before the introduction of the idea of the subroutine.

TEACH

- The major resource we had available to us in the summer of 1967 was the Compatible Time Sharing System (CTSS)
  - It includes a large number of typewriter consoles
  - an IBM 7094 computer
  - and a **generous** set of software facilities.
    - Of these software facilities, *the most important to us were the CTSS file system*
    - a good algebraic language (MAD)
    - a list-processing language (SLIP)
    - facilities for typewriter-computer communication.



Summary of Pre-LLM cITS

- Tedious to develop
- Expensive
- Not Scalable across task types and domain
- Somehow Robust
- Total Control on development process
- Partial Control on behavior

LLMs for Conversational ITSs

- **Did we solve the problem?**
  - **Yes**
    - Sonkar and colleagues (2023) present the LLM-based SPOCK, a conversational-based tutoring system for biology
    - indicate perfect (100%) or nearly perfect performance (upper 90s), depending on the performance criteria, when reporting the ability of LLMs to generate tutorial dialogues
  - **No**
    - LLMSs Get Lost In Multi-Turn Conversation (Laban et al., 2025)
      - all the top open- and closed-weight LLMs exhibit significantly lower performance in multi-turn conversations than single-turn, with an **average drop of 39% across six generation tasks**
      - LLMs often make assumptions in early turns and prematurely attempt to generate final solutions, on which they overly rely
      - *when LLMs take a wrong turn in a conversation, they get lost and do not recover*

iCODE: Adaptive Scaffolding of Students' Code Comprehension

Why iCODE?

- ***Reading code is the most time-consuming activity during software maintenance, consuming 70% of the total lifecycle cost of a software product***
- ***Code reading is even more important in the LLMs era***
  - ***now even more important given the LLMs can generate/draft code***

High vs. Low Cohesion Text

- Example A:
  - *George got some beer out of the car. The beer was warm.*
- Example B:
  - *George got some picnic supplies out of the car. The beer was warm.*

High vs. Low Cohesion Code

```
# items
a = {"Mango":3,"Banana":5,"Apple":2,"Orange":2,"Peach":6}
# total
b = 0
# count
for c, d in a.items():
    b += d
# display
print("The total Number of Fruits in the Basket is " + str(b))
```

```
# IMPORTANT
# Initialize the total number of fruits to count from 0
totalNumberOfFruits = 0

# Create a dictionary with types of fruit and how many of each there are
fruitsDictionary = {"Mango":3,"Banana":5,"Apple":2,"Orange":2,"Peach":6}

# Loop through the dictionary and add up the counts of each fruit to get the total
fruitNames = list(fruitsDictionary.keys())
countList = list(fruitsDictionary.values())
for currentCount in countList:
    totalNumberOfFruits += currentCount

# Print the total number of fruits
print("The total Number of Fruits in the Basket is ")
print(str(totalNumberOfFruits))
```

8 manipulations for code cohesion

- Comment Helpfulness
- Identifier Meaningfulness
- Syntax Highlighting
- Vertical Spacing
- **Argument Overlap**
- Lengthiness
- Signaling the Beacon
- ✨ Super Combination ✨

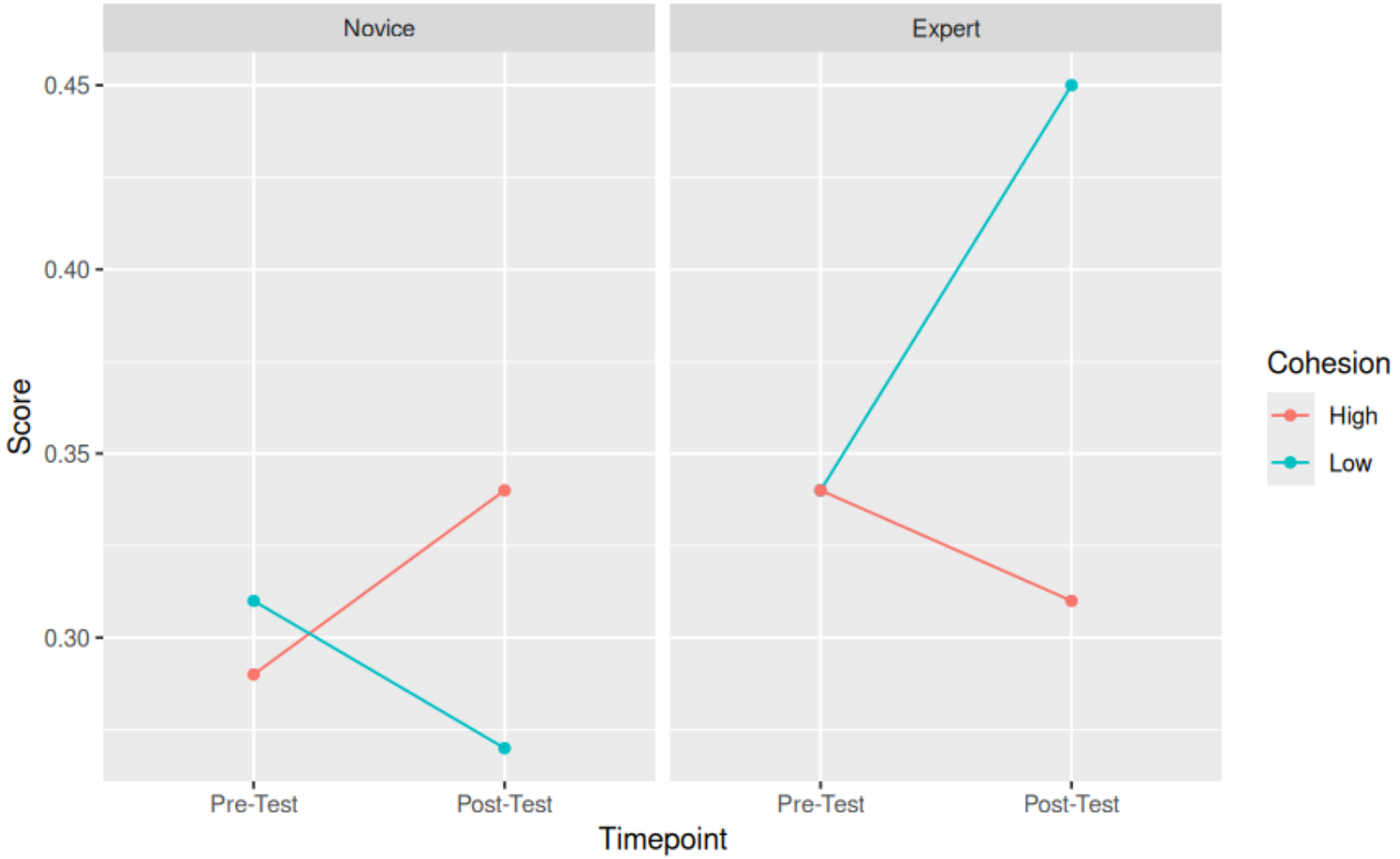
high

```
counts = list(fruits.values())
for count in counts:
```

low

```
counts = list(fruits.values())
message = "The total Number of
    Fruits in the Basket is "
for count in counts:
```

Reverse Cohesion Effect



Evidence for the Potential of LLMs to alleviate the ITS authoring bottleneck

- Use case:
  - iCODE, a code comprehension tutor
- Domain Modelling:
  - Infer a domain model, i.e., the set of knowledge components
    - KC extraction: precision=60s-70s%, recall=20s-70s% (at rank 5, 10, and 15)
  - Generate code examples and explanations linked to target knowledge components
    - Highly accurate (upper 90%) for intro-to-programming code examples
    - LLMs cannot simply generate code explanations that fit a/our theory-driven code explanation framework
- Learner Model
  - Auto-assessment of student responses, i.e., explanations of code in our case
  - GPT-4 leads to best results with an accuracy of 94% for the semantic similarity method compared to the best holistic method result which was for GPT-3.5 with an accuracy of 48%
- Pedagogical Model
  - Hint generation

Neuro-Symbolic AI for Math Tutoring



Vasile Rus



Deepak Venugopal



Stephen Fancsali

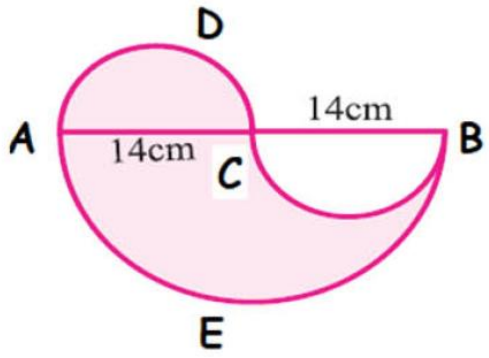


Steve Ritter

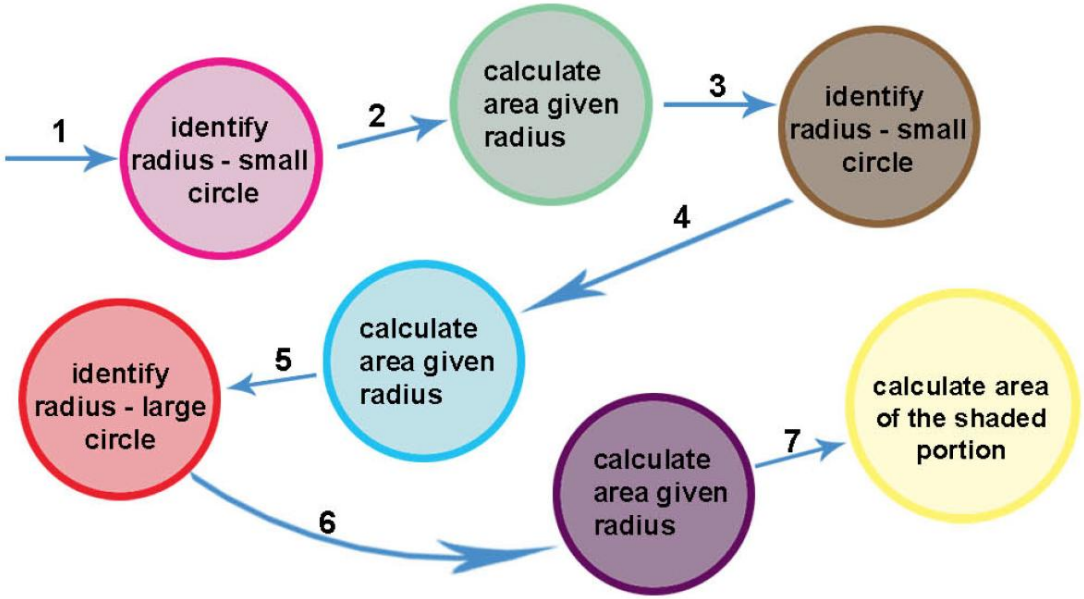
This research has been supported by:

ASTRA: AI-based Strategy Analysis

How do students solve math problems?



Find area of shaded portion



Goal: Learn **Large Strategy Models** for Math learning from educational datasets

MATHIA
(formerly known as Cognitive Tutor)

Student-directed problem solving

Active

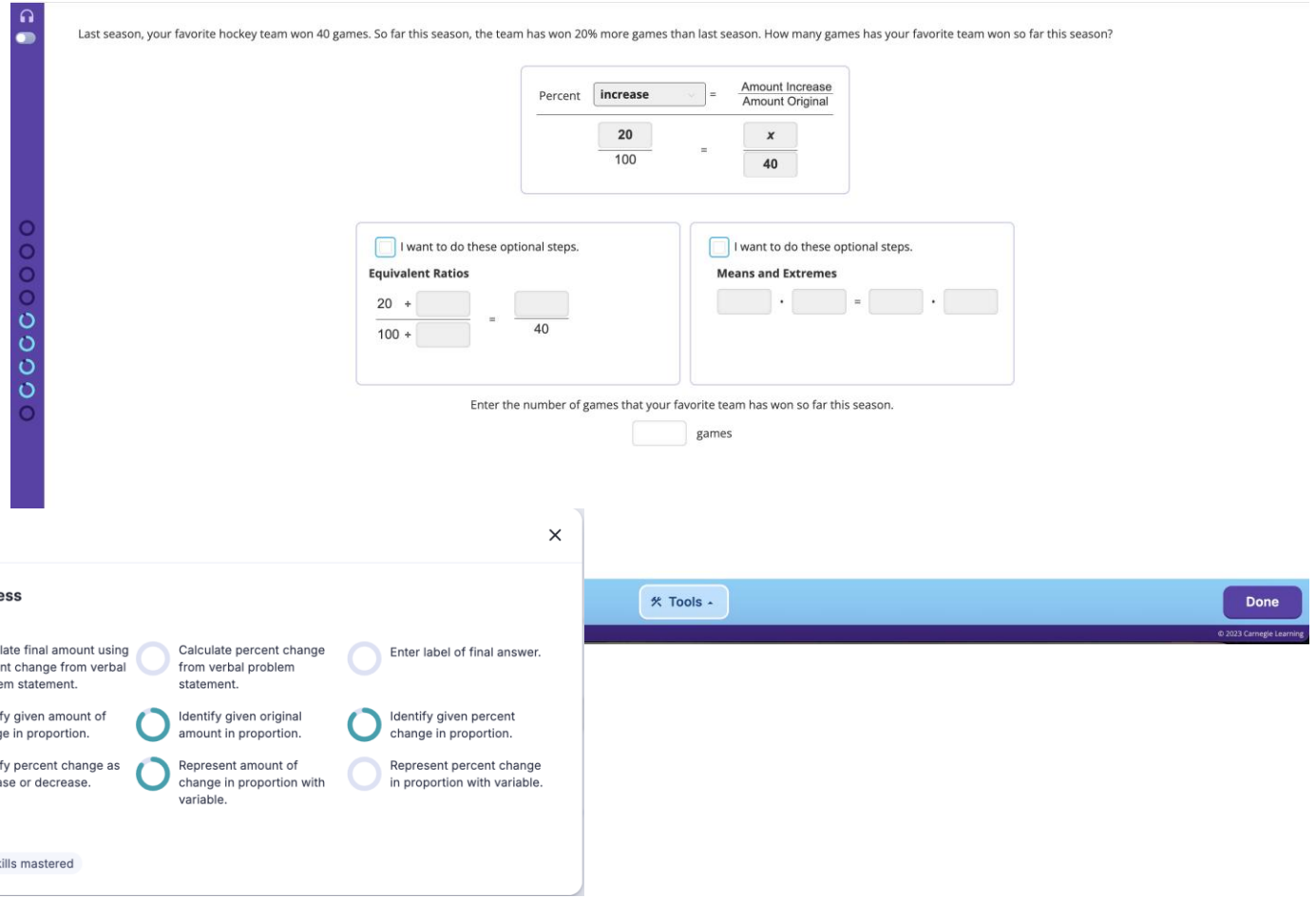
- Students are given problems to solve
- MATHia watches what they do

Relevant

- Students solve multi-step, real-world problems
- Feedback is personalized and contextual

Adaptive

- Knowledge component modeling determines which problems to present
- Students progress based on mastery



Student Strategy Prediction using a Neuro-Symbolic Approach

- Using Markov Logic Networks to learn a representation only
  - MLN inference/learning algorithms cannot scale up to large datasets and perform extremely poorly in such cases
- MLNs formulas specify relationships/connections in the MLN graph between different entities in our dataset (e.g. problems, students, etc.) while the actual learning and predictions are performed by an LSTM model.

$$\begin{aligned}\texttt{Student}(s) \wedge \texttt{Problem}(p_1) \wedge \texttt{Problem}(p_2) \wedge \\ \texttt{PHierarchy}(p_1, h) \wedge \texttt{PHierarchy}(p_2, h) \wedge \texttt{KC}(s, p_1, t_1, k) \\ \Rightarrow \texttt{KC}(s, p_2, t_2, k)\end{aligned}$$

Student Strategy Prediction using a Neuro-Symbolic Approach

Unit	LSTM-NS-Random (%)	LSTM-NS-NaiveGroup (%)	LSTM-NS-Clustered (%)	LSTM-NS-Adaptive (%)
PROBABILITY	67.7	53.48	**77.45**	70.73
FRACTION-OPERATIONS-1	74.95	78.23	84.02	**90.49**
PERCENT-CONVERSION	99.01	99.18	88.6	**99.5**
SCI-NOTATION	66.67	69.39	69.9	**70.16**
PICTURE-ALGEBRA-2	90.23	**96.17**	93.29	95.17
RATIONAL-NUMBER-OPERATIONS	30.76	41.85	40.28	**66.13**
INTEGERS	87.47	78.83	93.52	**96.78**
ORDER-OF-OPERATIONS	60.03	50.35	45.53	**69.43**
ONE-STEP-EQUATIONS	**66.6**	53.36	58.86	58.06
TWO-STEP-EQUATIONS	**63.13**	63.03	59.42	61.5

PreTraining-FineTuning A BERT-like Model

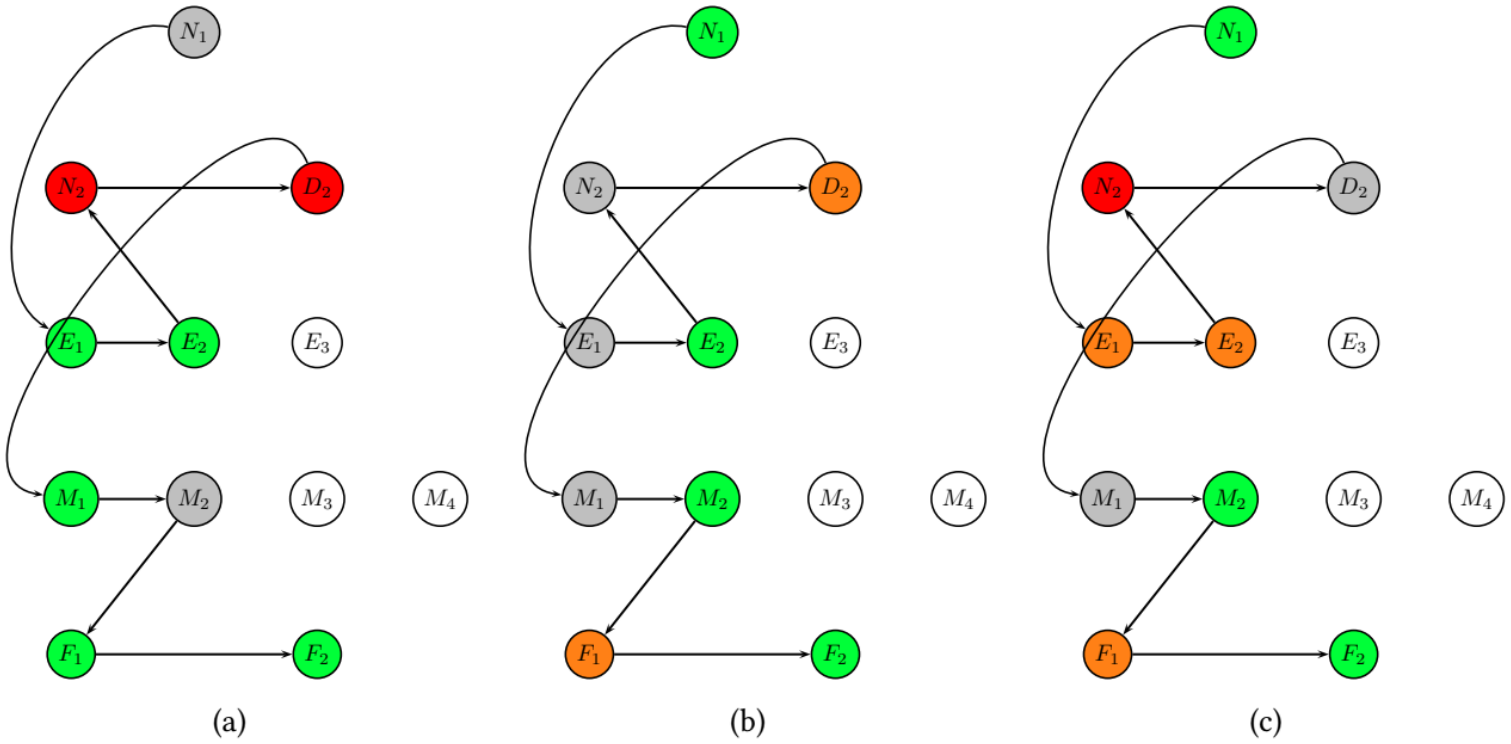
- Pre-training - predict the tokens at the masked positions from their embeddings using a softmax function

$$\ell(\Theta) = \sum_{i=1}^{N} \sum_{j=1}^{K} - \sum_{v} y_{ij}^{v} \log P_{\theta}(y_{ij}^{v} | \mathbf{X}_i)$$

- Fine-tuning on student success prediction task by minimizing the cross entropy loss

$$\ell(\hat{\theta}) = - \sum_{i=1}^{N} y_i \log(\hat{P}_{\theta}(\mathbf{X}_i) + (1 - y_i) \log(1 - \hat{P}_{\theta}(\mathbf{X}_i))$$

Masked Language Model (BERT-MLM)



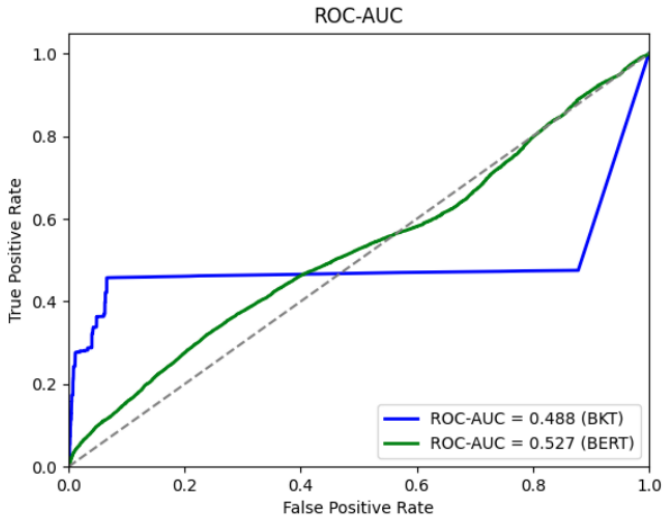
BERT-MLM

- The training is performed using the cross-entropy loss, where we predict the tokens at the masked positions from their embeddings using a softmax function.
- The loss that is minimized is:

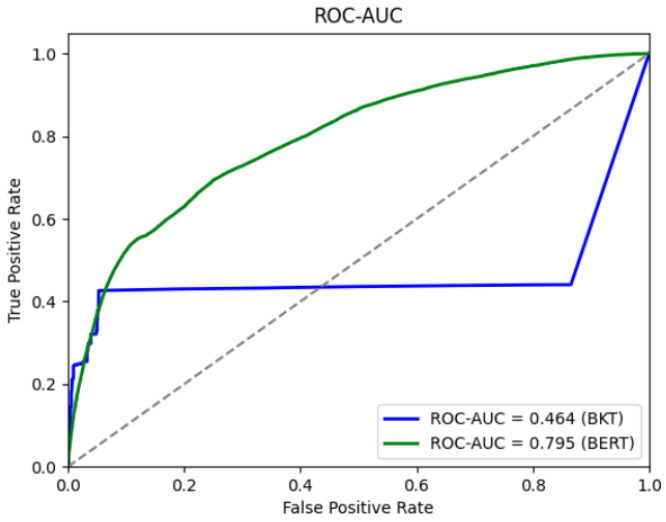
$$\ell(\Theta) = \sum_{i=1}^{N} \sum_{j=1}^{K} - \sum_{v} y_{ij}^{v} \log P_{\theta}(y_{ij}^{v} | \mathbf{X}_i)$$

- Architecture: L = 4, H = 64, and A = 8. (L=transformer blocks, H=size of hidden representation, A=attention heads)

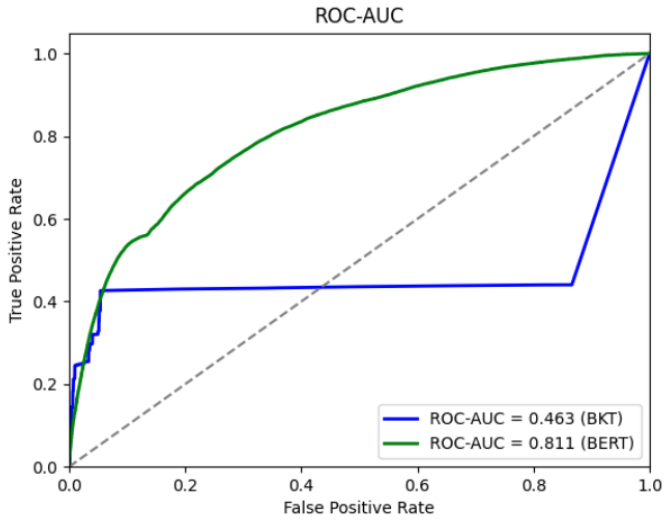
RESULTS



(a) 1% FT



(b) 5% FT



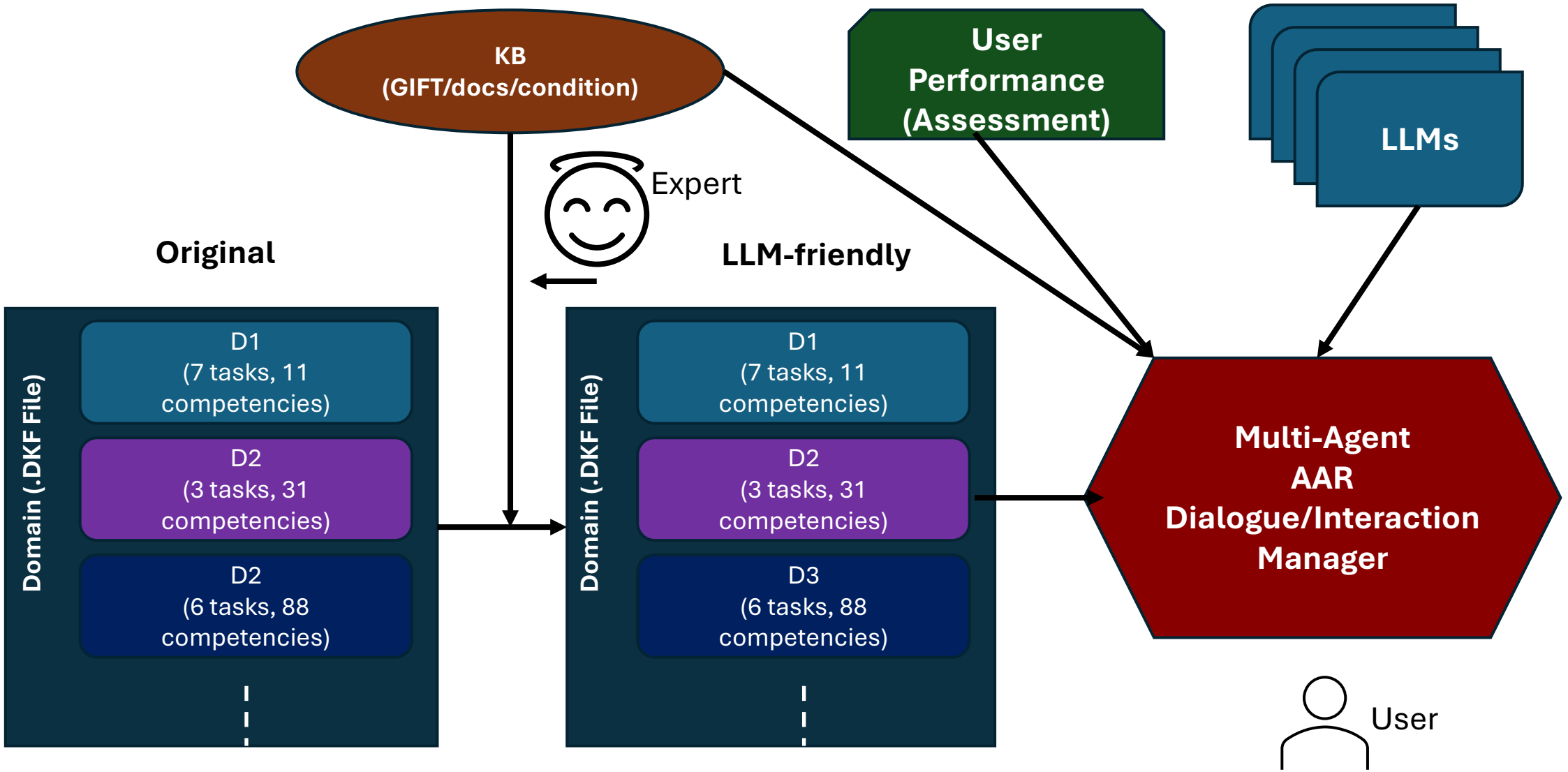
(c) 10% FT

LLM-based AAR

Overview

- General, Cross-domain/domain-**Scalable** Process
  - Individual level AAR
- Multi-agent dialogue/interaction manager that uses LLMs for specific tasks and micro- and short-dialogues
  - Cost-optimized by using different versions of the same LLM
  - Response time/Delay optimized
- Other Features
  - Stored, playable/playback demo dialogues
  - Synthetic data/user generation
  - Sandbox
- Next steps
  - Testing on more realistic user inputs
  - Evaluation on some sizeable dataset
  - Refinements and extensions

General, **Scalable** Process for Individual level AAR



Dialogue/Interaction Manager

- LLMs called upon for specific tasks/steps with significant prompt engineering which involves dynamic context update
  - Assessment of user input
  - Generation of user responses (synthetic data)
  - Generation of next dialogue move
  - Handling of the micro- and short-dialogues related to a particular competency
- LLMs can called upon for task/subtask or step/substep decomposition
  - Too risky at this time
  - It could raise the complexity of the design and testing process

LLMs

- LLMs are a refreshing new technology
- Turned many things upside down
  - Language generation seems easier than language understanding
  - Good at higher levels in Bloom's taxonomy, e.g., synthesizing/summarizing.
- Swiss-knife kind of tool: good (far from perfect) at many things/tasks
- Generally, good at one-turn interactions for general or well-known or synthesizing requests, i.e., one Q-A/P-R interaction or micro-dialogue
  - Approximate retrieval
- Mixed evidence and performance so far

LLMs

- **Elusive** Intelligence!
  - Sometimes it’s there, sometimes it’s not
- **Bad calculator**
  - Better know math if using one
  - If used for learning, better have a math expert/instructor nearby
- Need improvement in
  - **Robustness**
  - **Reliability**
  - **Controllability**

Outline

- Overview of the AIED landscape [*5 minutes*]
- Historical Perspective [5 minutes]
- Three Examples
  - iCODE: Adaptive Scaffolding of Learners' Code Comprehension Processes [*5 minutes*]
    - Funded by Department of Education [$2 million]
  - AAR-Tutor: An After Action Review Tutor [*5 minutes*]
    - Funded by the Department of Defense [$1.4 million]
    - Part of the U.S. Army's Generalized Intelligent Framework for Tutoring (GIFT – gifttutoring.org)
  - Neuro-Symbolic AI for Math Tutoring [*5 minutes*]
    - *Investigating techniques that couple Markov Logic and Deep Learning with applications to discovering strategies to improve STEM learning* – funded by NSF [$400k]
    - *ASTRA: AI-based Strategy Analysis* – funded by Gates Foundation [$500k]
- **Discussion** [*5 minutes*]

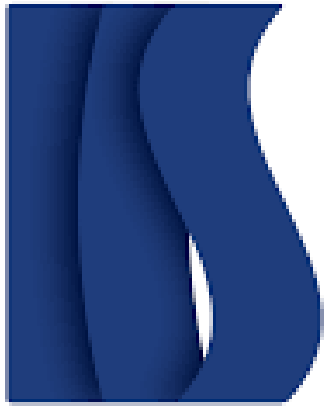


Discussion

- Reading vs. Writing First Approach in CS Education

- Are LLMs Good for Learning? (i.e., human learning)

AI in Education



Institute for Intelligent Systems

THE UNIVERSITY OF MEMPHIS

Vasile Rus, PhD

The University of Memphis

vrus@memphis.edu

THE UNIVERSITY OF MEMPHIS®

Additional Slides

Role Assignment

- ***You are a tutor!***
- Magically, LLMs will act like a conversational tutor

Three Approaches to Using LLMs for ITS Authoring/Development

One-Page Prompt/ Role-Assignment

- **One-page prompt**
  - One-page full-specification of an tutoring/ITS
  - ***LLM driving the interaction***
- **Role-assignment** followed by instructional help request
  - No detailed specification of tutoring/ITS
  - ***The user/learner driving the conversation***
- Over-reliance on LLMs' capabilities
- Over-reliance on an initial, overarching prompt

Emulating ITSs

- Inducting conversational ITSs that follow the AutoTutor's Expectation-Misconception Tutoring framework
- Over-reliance on LLMs' capabilities
- Hard to assess the LLMs' capabilities on specific tasks
- Existing trials are not really ITSs but rather enhanced e-books

Task-Specific Use

- Point-wise, task specific use of LLMs
- Intense, task-specific prompt engineering and/or fine-turning
- Specific task modelling research needed
- In-depth understanding of LLMs' capabilities to help with various ITS items and components
- Rely on existing ITS platforms

Lessons Learned

- the exact modelling of a target task when using LLM-based methods is very important
- there is a wide range in performance across various LLMs with GPT-4 leading to best results in most cases
- Large Language Models (LLMs) perform comparably well, in particular GPT models, compared to fine-tuned encoder-based models
- LLMs are a general model that works as well as more specialized solutions on many tasks
  - a Swiss knife kind of tool
  - while it does not completely solve many of these tasks it provides state-of-the-art solutions.