Leveraging Rationales to Improve Human Task Performance

Paper #1345

ABSTRACT

Machine learning (ML) systems across many application areas are increasingly demonstrating performance that is beyond that of humans. In response to the proliferation of such models, the field of Explainable AI (XAI) has sought to develop techniques that enhance the transparency and interpretability of machine learning methods. In this work, we consider a question not previously explored within the XAI and ML communities: *Given a computational system whose performance exceeds that of its human user, can explainable AI capabilities be leveraged to improve the performance of the human?* We study this question in the context of the game of Chess, for which computational game engines that surpass the performance of the average player are widely available. We introduce the *Rationale-Generating Algorithm*, an automated technique for generating rationales for utility-based computational methods, which we evaluate with a multi-day user study against two baselines. The results show that our approach produces rationales that lead to statistically significant improvement in human task performance, demonstrating that rationales automatically generated from an AI's internal task model can be used not only to *explain* what the system is doing, but also to *instruct* the user and ultimately improve their task performance.

CCS CONCEPTS

• **Human-centered computing** → **Interaction paradigms**.

KEYWORDS

Explainable AI, Machine Learning

ACM Reference Format:

Paper #1345. 2019. Leveraging Rationales to Improve Human Task Performance. In *IUI '20: ACM International Conference on Intelligent User Interfaces, March 17–20, 2020, Cagliari, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Machine learning (ML) systems across many application areas are increasingly demonstrating performance that is beyond that of humans. From games, such as Chess [19] and Atari [24], to high-risk applications, such as autonomous driving [29] and medical diagnosis [20], human users are becoming increasingly surpassed by, and reliant on, autonomous systems. In response to the proliferation of such models, the field of Explainable AI (XAI) has sought to develop techniques that enhance the transparency and interpretability of machine learning methods. XAI approaches have included taking advantage of intrinsically interpretable models (e.g., decision sets) [21, 22], as well as developing interpretable approximations to explain the behavior of non-interpretable black-box models (e.g., decision tree approximations of deep neural networks) [4, 25]. The vast majority of XAI research has focused on expert users, for example, medical personnel evaluating the decision-making capability of an automated diagnosis system [2, 25].

In this work, we consider a question not previously explored within the XAI and ML communities: **Given a computational system whose performance exceeds that of its human user, can explainable AI capabilities be leveraged to *improve* the performance of the human?** In other words, can the increasingly powerful machine learning systems that we develop be used to, in turn, further human capabilities? Within the context of XAI, we seek not to explain to the user the inner workings of some algorithm, but to instead communicate the rationale behind a given decision or choice. The distinction between *explanation* and *rationale* has been previously introduced by Ehsan et al. [12], who defined explanations as a way to expose the inner workings of a computational model through any communication modality (e.g., visual heatmap [13]), often in a way that is accessible only to experts. Rationales, on the other hand, are defined as natural language explanations that do not literally expose the inner workings of an intelligent system, but instead provide contextually appropriate natural language reasons. These natural language reasons are accessible and intuitive to non-experts (e.g., "I had to go forward to avoid the red vehicle."), facilitating understanding and communicative effectiveness. Within the context of their work, Ehsan et al. introduced a computational method for automatically generating rationales and validated a set of human factors that influence human perception and preferences (i.e., contextual accuracy, intelligibility, awareness, reliability, strategic detail) [12].

In this work, we explore whether providing human users with a *rationale* of an intelligent system's behavior can lead to improvement in the user's performance. We study this question in the context of the game of Chess, for which computational game engines that surpass the performance of the average player are widely available. Our work makes the following contributions:

- We introduce the *Rationale-Generating Algorithm (RGA)*, an automated technique for generating rationales for utility-based computational methods.
- We study two variants of our approach, one that takes advantage only of the system's knowledge (RGA), and a second (RGA+) that additionally incorporates human domain expert knowledge.

We evaluate both techniques in a multi-day user study, comparing against two baselines to measure human learning performance. The results demonstrate that our approach produces rationales that lead to improvement in human task performance in the context of chess endgames. Using winning percentage and percentile rank of player moves as a measure of performance, we observe that the inclusion of domain expert knowledge (RGA+) significantly improves

IUI 2020, March 17–20, 2020, Cagliari, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

human task performance over both baselines when compared to non-rationale baselines. Additionally, user self-reported performance ratings also show that rationale-based interfaces lead to greater perceived user understanding of the task domains than non-rationale baselines. In summary, our approach is the first to demonstrate that rationales automatically generated from an AI's internal task model can be used not to just *explain* what the system is doing, but also to *instruct* the user and ultimately improve their task performance.

2 RELATED WORKS

Traditionally, in the XAI community, an interpretable model can be described as one from which an AI expert or user can deduce model performance based on given inputs [12]. The methods of interpretability can vary based on the complexity of a model and a wide range of survey papers summarize the different XAI models currently developed [2, 25, 33]. While some of the existing models are inherently interpretable, suitable for model-intrinsic decision-making [7, 22], other complex models need model-agnostic approaches for interpretability [15, 30–32]. Despite the differing approaches for interpretability, XAI models have the common motivation of improving human understanding of AI systems and building trust with these systems [16, 17].

One existing method for developing interpretability is to use intrinsic models in which interpretability is embedded inside the model. In [22] a Bayesian Rule List (BRL) is created to produce posterior distributions over permutations of *'if, then, else'* rules for a single prediction classification. Since these decision lists are intrinsically understandable, using them as the basis of BRL is successful in developing interpretability. However, the accuracy of BRL depends highly on the Bayesian prior favoring concise decision lists and a small number of total rules, limiting the applicability of this approach. To extend the applicability of model-intrinsic methods beyond the scope of concise decision lists, a generative additive model (GAM) is created to provide both high accuracy (better than random forest, logitboost and SVMs) and high interpretability for a single prediction classification [7]. To achieve model understanding, GAM creates a visual representation of the pairwise factors that result in a prediction and provides an ability for modular testing. [7] refers to modular testing as allowing model-experts to easily remove and insert individual factors or pairwise factors to examine their effects on a prediction. While methods such as GAM provide interpretability for regression and some classification models, the intrinsic nature of its model makes it hard to provide interpretability for more black-box models which need model-agnostic implementations of interpretability.

An alternative to the model-intrinsic approach described above is a model-agnostic method for interpretability. Model-agnostic methods are a significant focus within the XAI community since model-agnostic methods provide high performance accuracy, but do not allow for inherent decoding and visualization of model prediction. Most model-agnostic methods produce interpretability by developing surrogate models, post-hoc implementations of interpretability derived from inherently interpretable models. In [15] an ad-hoc genetic algorithm is used to generate neighbor nodes for a specific local instance, which is then trained by a decision tree classifier to generate a logic rule. This logic rule represents the path in the decision tree that explains the factors that lead to a prediction. Additionally, [32] uses inherently explainable decision tree models to learn CNN layers in order to provide CNN rationales at the semantic level. [32] defines a CNN rationale as an ability to quantify the set of objects that contribute to a CNN prediction from prediction scores. [31] also uses the concepts of explanatory graphs to visually reveal the hierarchy of knowledge inside CNNs in which each node summarizes the knowledge in the feature maps of a corresponding conv-layer. However, the decision tree rationales and explanatory graphs only provide quantitative distributions and visualizations to domain-experts, and are not interpretable to general users without domain knowledge. Moreover, the complexity of these decision trees vary greatly based on the complexity of the model, sometimes no longer staying interpretable due to a large node space. Thus, [30] creates a tree regularization penalty function that helps produce interpretability of complex models by using moderately-sized decision trees. Altogether though, the focus of the intrinsic and agnostic methodologies remain largely focused on giving insight into the inner workings of the AI systems, rather than providing humanly understandable rationales that extend beyond domain-expert usability.

The HCI community acknowledges the gap between XAI and the human usability of XAI, communicating that the explanations from AI and ML communities do not yet possess a large-scale efficacy on human users [1]. To address the importance of user-friendly explanations for any user, not only domain-experts, researchers in HCI have developed context-aware rules to focus interest on easy-to-use interfaces that are aware of their environment and context of use [11]. However, many of these existing context-aware rules are developed as frameworks to aid decision-making in domain-specific applications such as smart homes [5, 10] and the office [9], instead of utilizing a model-agnostic approach encouraged by the AI/ML community. In addition to the context-aware rules used above, educators have focused on intelligent tutoring systems (ITS) [6, 18, 23] to generate explanations for learning. While we similarly seek to improve the performance of a non-expert user, our work differs from tutoring systems in that the rationales produced by our system are generated based on and reflect an AI's internal computational model, not a prescribed curriculum.

To address a more generic solution for generating human understandable explanations from AI/ML produced models outside of a specified curriculum, [12] develops human understandable rationale in the context of the game Frogger. The rationale generation uses natural language processing to contextualize state and action representations for a reinforcement learning system and studies the human factors that influence explanation preferences. The metrics for measuring interpretability of the rationales are primarily focused on perceived understanding of the rationale. In contrast, we generate rationales and measure task performance and self-perceived task performance improvement, as opposed to surveying understanding of a given rationale.

3 RATIONALE GENERATING ALGORITHM

XAI systems are often used to aid users in the decision-making process for some task, such as medical diagnosis [20]. In this work, we introduce the Rationale Generating Algorithm (RGA), which

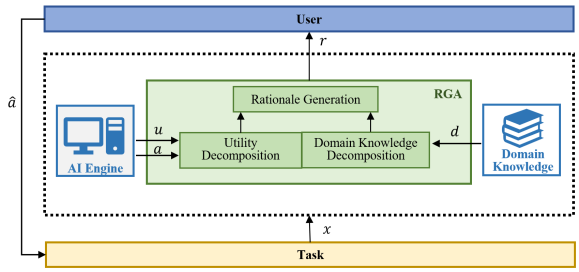


Figure 1: System overview diagram showing the interaction between a user, AI agent and RGA, in which x represents state, a represents action, u represents utility function, d represents domain knowledge and r represents rationale.

provides the user with rationales designed to aid the user's decision-making process while also increasing their *understanding* of the underlying task domain. Figure 1 presents an overview of the RGA pipeline. For a given task, we assume an expert AI Engine is available that takes in the current task state x, and outputs a recommended action a and its associated utility function u[1]. Within RGA, the utility function is decomposed to identify the *most significant factors* contributing to the selection of a over alternate actions. These factors take the form of variables, typically not very interpretable in their original form (e.g. 'PassedPawns = 0.7'). Additional factors that support decision making may also be obtained from provided expert domain knowledge (see below). While a utility function may be made up of dozens of variables, RGA selects only the top k factors, which are then used to generate a human-readable rationale. Note that [14] has shown that in the context of an explainable planner, generating justifications to explain both good and bad action choices leads to a more robust explainable system. Motivated by this, RGA is able to generate rationales for both positively and negatively contributing actions.

Algorithm 1 further details the manner in which RGA is implemented and a final rationale is chosen. Variables P and D (lines 1-2) store the name of each decision factor contributing to the selection of a given action, and its associated weight for utility-based (w_p) and domain-based (w_d) knowledge, respectively. Given the utility function u used to select the given action a, RGA first decomposes u to obtain the list of all factors involved in the determination of the action and their contributing normalized weight. The resulting list is stored in P (line 3).

RGA next optionally processes the domain knowledge, if it is available (lines 4-6). We define domain knowledge as any externally available data, typically encoded a priori by a domain expert [28]. The domain knowledge is encoded as a supplemental utility function d and used to supplement RGA with information that is not encoded in u but might be helpful to include in a rationale. For example, within the Chess domain we found that the default u generated by

[1]The presented variant of RGA requires a normalized utility function to generate rationales. Our objective in this work is to evaluate the effect of rationales on user performance, thus we did not focus on developing a fully model-agnostic rationale generation technique. In general, approaches such as [12] can be used to generate rationales for models that do not incorporate a utility function

Algorithm 1 generateExp(u, a, d)

Input : u - utility function, a - selected action, d- domain knowledge

Output : r - rationale

1: P=[$name_p$:{}, w_p:{}]
2: D=[$name_d$:{}, w_d:{}]
3: P= decomposeUtility(u)
4: **if** d is not Ø **then**
5: D = decomposeDomainKnowledge(d)
6: **end if**
7: $Factors = P \cup D$
8: $Factors$.sortByWeight()
9: **for** f in $Factors[i:k]$ **do**
10: **if** isPositive(a) **then**
11: r = genPos(f.name, a)
12: **else**
13: r = genNeg(f.name, a)
14: **end if**
15: **end for**
16: **return** r

our game engine did not include a variable for 'Checkmate' (an important winning condition in chess), whereas providing information about a checkmate within a rationale would likely be helpful to the user. As a result, we include the concept of domain knowledge, and in our experimental section compare RGA performance with and without domain knowledge. If domain knowledge is present, RGA decomposes it similarly to the utility function into a list of factor names $name_d$ and their associated weights w_d (line 5).

Once decision factors from the utility function and the domain knowledge are identified, the weighted lists for both sets of factors are merged and sorted by weight in descending order (lines 7-8). The top k factors are then used to generate a rationale, with appropriate wording being used dependent on whether action a is being recommended as a desirable action or not (lines 10-14). If a represents a positive action, then a positive rationale is scripted, explaining how each of the k features aid in task completion (e.g.'A boat can help cross the river because it floats on water'). If a represents a negative action, then a negative rationale is scripted, explaining how each of the k features inhibit task completion (e.g. 'A car cannot help cross the river because it does not float on water').

Once a rationale is returned by RGA, we display it to the user along with the recommended action a selected by the AI Engine. The user may then leverage the rationale, and their own knowledge of the domain, to decide whether to perform a or to select a different action. The user's final selected action $\hat{a}$, which may be the same or different from a, is then applied back to the task domain.

4 RGA AND CHESS

To measure the effect of human-understandable rationales on task performance, we apply RGA to the game of chess, specifically focusing our attention on endgame configurations, defined by no more than 12 pieces on the board. Chess is a good application for RGA as

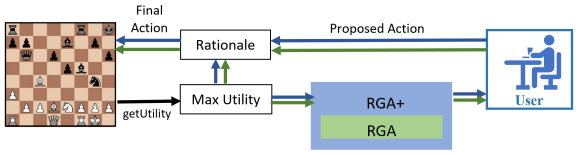


Figure 2: An overview of the application of RGA to the domain of chess, highlighting the two variants of RGA.

the utility function for chess is complex in nature, involving many parameters, and the game environment continuously changes over time. We focus specifically on endgame scenarios in which decision making is crucial to an outcome, and there is a relatively small amount of moves left. With endgame scenarios, we are able to analyze the effects of RGA through a measured experimental design.

For this research, we use the the utility function from the open source Chess AI Engine, Stockfish [27], and utilize RGA to generate human-understandable rationale for both an optimal and any non optimal moves taken. This section discusses RGA within the chess domain.

4.1 Chess Utility Function

The utility function from Stockfish includes over 100 utility factors that are combined in a multi-level hierarchical fashion. While all of these factors can be crucial to explaining different moves within a chess board configuration, only a subset of these main factors are useful for explaining moves in context of endgame configurations [3]. The relevant endgame utility factors are determined after sorting all factors by maximum weight (line 8 of Algorithm 1). They include but are not limited to: 'Mobility', 'KingDanger', 'King', 'HangingPiece', 'PawnPromotion', and 'Passed'.

As a brief summary, 'Mobility' refers to the number of legal moves a player has for a given position–the more choices a player has, the stronger their position on the board. Also for endgames, the king is a powerful piece and is preferred to keep in the center of the board due to its limited range; 'KingDanger' is the highest weighted feature that affects the 'King' score. Additionally, 'Threats' play a huge role in the outcome of an endgame, and are primarily represented by crucially attacking pieces such as rooks and kings, and by 'HangingPiece' which refers to weak enemy pieces not defended by the opponent. 'Passed' refers to the concept of 'Passed Pawns' which checks to see if a pawn can continue to safely advance to the other side of the board. 'Passed', includes the feature 'PawnPromotion' which details when a pawn has reached the opponent's top rank, allowing the player to switch its pawn out for a queen, rook, bishop or knight. These described utility factors represent the utility factors that we deemed appropriate from the Stockfish utility function, based on their weights, to use in justifying a non-optimal or optimal move choice.

4.2 Data Pre-Processing

To meet RGA's requirement of having a normalized utility function input, we standardize the Stockfish utility function using Z-score scaling. For Z-score scaling, we perform a heuristic ad-hoc analysis by collecting average standard deviations and means for each relevant factor over several game configurations. Specifically, we collect 120 game configurations using a random FEN generator. To ensure completeness, we represent an equal distribution of game configurations within the range of 2-32 pieces. We started with 30 random game configurations, and doubled them twice, until the change in both average standard deviation and mean for each factor was negligible.

4.3 RGA In Chess

In the context of chess, RGA generates human understandable rationales with the objective of checkmating the opponent. The Stockfish AI engine generates the recommended action for the current board state x, and its associated utility u. Figure 2 depicts the overall interaction between the Chess Engine, RGA, and the user. The $getUtility$ function obtains all relevant factors, actions and utilities for a given board configuration. By sorting these utilities, $MaxUtility$ finds the highest utility u_{best} and corresponding optimal move, a_{best} which RGA uses to generate a rationale to the user to justify the optimal move he/she should take. Additionally, we detect non-optimal moves by comparing the user's proposed action to the set of all possible actions A; if the proposed action falls in the bottom 1/3 of A, then we use the factors in $Factors$ for such proposed action to generate a cautionary rationale justifying why a user should not make such move. The user can ultimately decide upon the final move $\hat{a}$, considering or disregarding the rationales produced by RGA. Figure 3 provides an example of a best-move and a non-optimal move explanation using both a possible Stockfish and domain-knowledge factor.

4.4 Domain Knowledge Factors

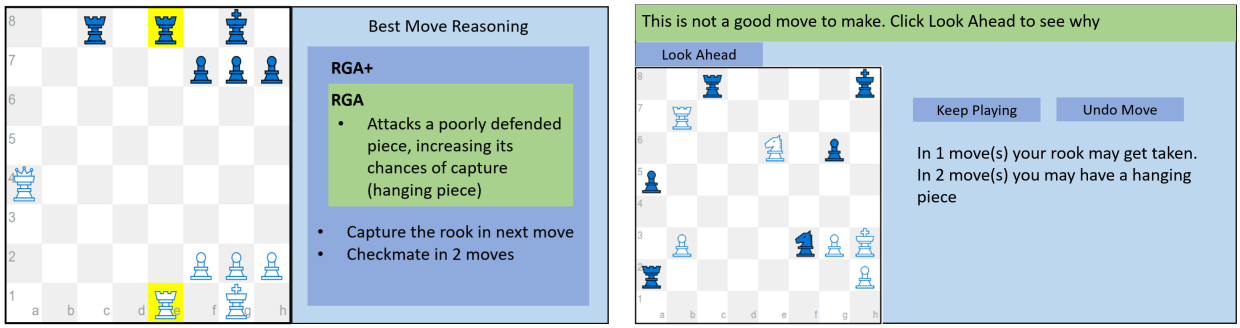
In the application of chess, we use $domainKnowledge$ to adds three important criteria that the Stockfish utility function does not explicitly represent: (1) explicit piece capture on next move, (2) check on next move, (3) checkmate on next or subsequent move. Considering the objective of chess, we weigh these additional domain-knowledge factors higher than those from the utility function. Shown in Figure 2, we distinguish RGA+ to be a superset of RGA. The overarching RGA+ denotes rationales that are reasoned from both domain knowledge and the chess utility function, whereas the RGA requires a utility function for rationale generation, but leaves domain knowledge inputs as optional to domain experts.

5 EXPERIMENTAL DESIGN

To evaluate the effectiveness of RGA in improving task performance, we conducted a four-way between-subjects user study in which participants took part in chess gaming sessions over three consecutive days. We selected a multi-day study design to ensure observation of longer-term learning effects. Given the complexity of chess, which requires an estimated 10 years [26] or 5000 hours [8] to master, we conducted our study using only simplified game scenarios consisting of *end games*.

The study design consisted of the following four study conditions, which determine what guidance was provided to the participant:

- **None (baseline)**: The player receives no hints or rationales. This condition is equivalent to practicing chess independently with no guidance. (Figure 4(a))



(a) (b)

Figure 3: Example rationales generated for a particular board configuration where (a) represents a best-move explanation and (b) represents a non-optimal move explanation.

- **Hints (baseline)**: The player receives a visual hint highlighting in color the best currently available move, as determined by the game engine utility function. No textual rationales are provided beyond the highlighting. This condition is equivalent to the hints system available in the vast majority of online and app-based chess programs. (Figure 4(b))
- **RGA**: The player receives a visual hint highlighting in color the best currently available move (as in Hints). Additionally, the system displays a textual rationale based the Stockfish utility function only. (Figure 4(c))
- **RGA+**: The player receives a visual hint highlighting in color the best currently available move (as in Hints). Additionally, the system displays a textual rationale based on both the Stockfish utility function and domain knowledge. (Figure 4(c))

The sections below further detail our evaluation metrics, study hypotheses, participant recruitment method, and study design.

5.1 Metrics

Each chess session consisted of *diagnostic games*, during which participants were evaluated on their performance and received no suggestions or hints, and *instructional games*, during which participants received guidance according to their assigned study condition. During diagnostic games, the following metrics were used to evaluate performance:

- **Win Percentage (*Win%*):** a metric commonly used in sports that takes into account the number of *wins*, *losses* and *ties*. In our domain, we additionally account for cases in which the maximum number of allowed moves has been reached[2], *maxmoves*, which are weighted the same as ties. The final win percentage is calculated as:

$$Win\% = \frac{wins + 0.5 \cdot ties + 0.5 \cdot maxmoves}{wins + ties + losses + maxmoves} \quad (1)$$

- **Percentile Rank (*Percentile*)**: a metric used to measure the distribution of scores below a certain percentage. In our domain, the distribution of scores is the move rating of all possible moves for a given board configuration. The move ratings are calculated by Stockfish. For each board configuration, we use the following formula to calculate the percentile rank of a chosen move k with N corresponding to all possible moves:

$$PercentileRank = \frac{100 \cdot (k-1)}{(N-1)} \quad (2)$$

Additionally, at the end of each study day, participants were given a short post-session questionnaire from which we obtain the following metric:

- **Perceived Performance (*SelfEval*)**: a metric that seeks to capture the participants' self-reported perceived progress toward learning chess. Perceived performance is measured using a 5-Point Likert Scale rating based on the question 'Do you believe your performance improved this session?' (1 = Strongly disagree, 5 = Strongly agree')

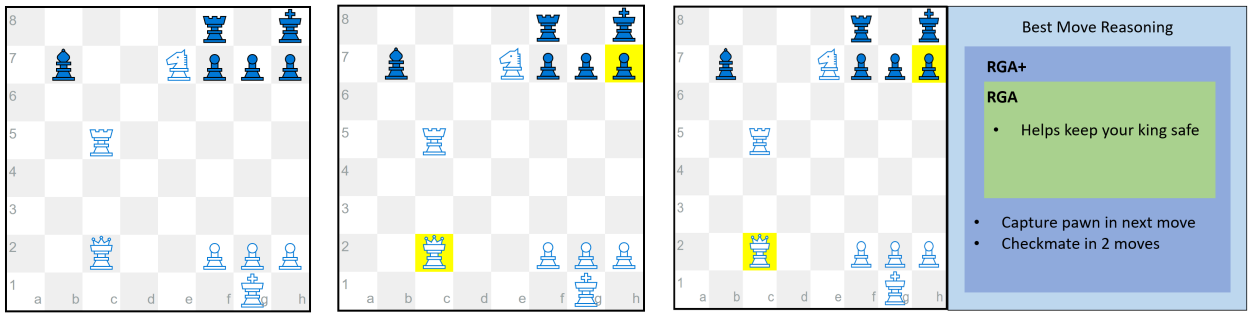
5.2 Hypotheses

We formulate the following hypotheses on the ability of interpretable rationales to improve participants' chess performance defined by the *Win%* and *Percentile Rank* metrics:

- **H1a**: Participants who received only utility-based rationales (RGA) will perform better than those who received no guidance (None).
- **H1b**: Participants who received only utility-based rationales (RGA) will perform better than participants who received only suggestions (Hints).
- **H1c**: Participants who received rationales that incorporate domain knowledge (RGA+) will perform better than participants who received no guidance (None).
- **H1d**: Participants who received rationales that incorporate domain knowledge (RGA+) will perform better participants who received only suggestions (Hints).
- **H1e**: Participants who received rationales that incorporate domain knowledge (RGA+) will perform better than participants who received only utility-based rationales (RGA).

Additionally, we hypothesize that participants' measure of their own perceived performance, evaluated by the *SelfEval* metric, will follow similar trends as above. Specifically:

[2]We limit the total number of moves to prevent novice players from moving around the board indefinitely.



(a) None (b) Hints (c) RGA (green) and RGA+ (blue)

Figure 4: Given a set game configuration, the three chessboards show the different user interfaces for the four experimental conditions where (a) represents the 'None' cohort, (b) represents 'Hints' and (c) represents the 'RGA+ and RGA' cohort.

- **H2a**: Participants who received only utility-based rationales (RGA) will have higher perceived performance ratings than those who received no guidance (None).
- **H2b**: Participants who received only utility-based rationales (RGA) will have higher perceived performance ratings than those who received only suggestions (Hints).
- **H2c**: Participants who received rationales that incorporate domain knowledge (RGA+) will have higher perceived performance than participants who received no guidance (None)
- **H1d**: Participants who received rationales that incorporate domain knowledge (RGA+) will have higher perceived performance than who received only suggestions (Hints).
- **H2e**: Participants who received rationales that incorporate domain knowledge (RGA+) will have higher perceived performance ratings than those who received only utility-based rationales (RGA).

5.3 Participants

We recruited 68 participants from Amazon's Mechanical Turk. Participants were required to demonstrate basic knowledge of chess by passing a short test verifying the rules of the game (e.g.'Which piece can move straight forward, but captures diagonally?'). Participants were also required to participate three days in a row, and to not already be expert players. Eight players were removed from the study for either not participating for three days or for winning every game (suggesting that they were experts to begin with). The final set of participants included 60 individuals (44 males and 16 females), who ranged in age from 18 to 54 (6 between 18-24 years, 31 between 25-34 years, 18 between 35-44, and 3 between 45-54 years). Participants were randomly assigned to one of the four study conditions. Each daily session took approximately 15-20 minutes, and participants were compensated $2.00, $4.00, and $6.00 on days 1, 2 and 3, respectively.

5.4 Study Design

The study consisted of three sessions performed on three consecutive days. In each session, participants played 9 games of chess: three diagnostic games, followed by three instructional games, followed by three more diagnostic games. The use of diagnostic games at the beginning and end of each session enabled us to study participant performance both across and within sessions[3].

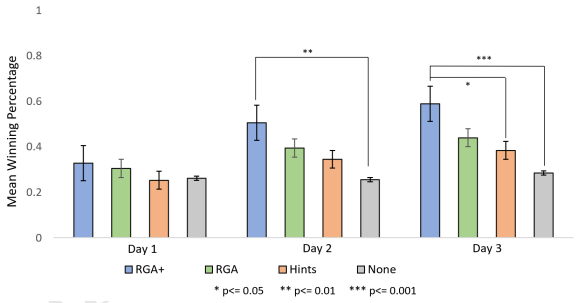


Figure 5: Average *Win%* of the experimental conditions.

The participant always played white, and the opponent black pieces were controlled by the Stockfish Engine AI, which always played optimally. For each board, the optimal number of player moves needed to win, O, was determined by the Stockfish Engine, and participants were limited to 10 moves during the game. Starting board configurations were obtained from a popular online learning website (https://lichess.org/), selected such that $O < 5$. All participants received the same boards in the same order to ensure uniformity; each board was unique and did not repeat. Players were allowed to make any legal chess move, and each game consisted of an average of 6 moves (SD=2.62). As a result, participants in the Hints, RGA and RGA+ conditions received approximately 18 move suggestions per day on average.

6 RESULTS

The participant performance data followed a normal distribution; as a result, we used ANOVA with a Tukey HSD post-hoc test to evaluate statistical significance across the experimental conditions with respect to the *Win%* and *PercentileRank* metrics. Additionally,

[3]Our analysis showed no significant trends for within-session performance differences, likely due to the short duration of the sessions. However, we do observe significant learning effects across sessions, as discussed in the Section 6.

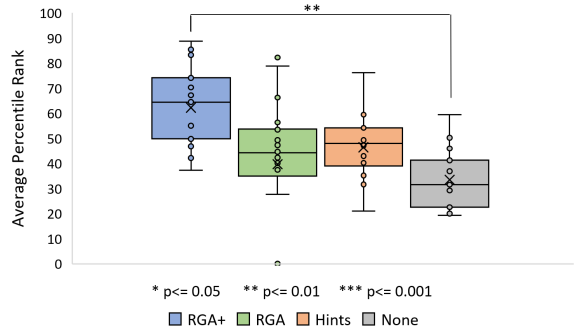


Figure 6: Average *PercentileRank* ratings of moves made by participants across all days.

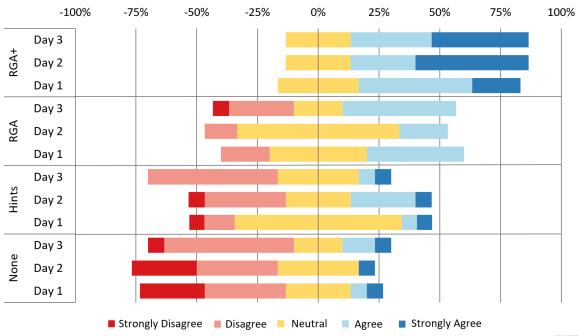


Figure 7: Survey data of participants' *SelfEval* across the experimental conditions.

Conditions	Day 1	Day2	Day3
RGA+ vs. RGA	NS	$p \leq 0.01$	$p \leq 0.05$
RGA+ vs. Hints	$p \leq 0.01$	$p \leq 0.01$	$p \leq 0.001$
RGA+ vs. None	$p \leq 0.001$	$p \leq 0.001$	$p \leq 0.001$
RGA vs. Hints	NS	NS	NS
RGA vs None	$p \leq 0.05$	$p \leq 0.01$	NS
Hints vs None	NS	NS	NS

Table 1: Mann-Whitney U test significance values for the *SelfEval* metric

we conducted a Mann-Whitney U test to analyze the Likert scale data for the *SelfEval* metric.

6.1 Participant Performance

Figure 5 presents the average participant win percentage (*Win%*) for each study condition over the three days. We observe that while no statistical differences are observed between conditions on the first day, the differences in performance grow on subsequent days. In particular we see the greatest rate of overall task improvement from Day 1 to Day 3 for the RGA+ condition. Our results indicate a correlation between the amount of explanation participants are given and their *Win%*, with more justifications leading to more wins. This is further supported by the results in Figure 6, which present the percentile rank (*PercentileRank*) of each participant's average move–with a percentile rank of 100 denoting the most optimal move and percentile rank of 0 denoting the least optimal move. Similar to Figure 5, we observe a correlation between the *PercentileRank* of move ratings and the amount of justifications provided, with the upper quartile for *PercentileRank* being higher with more explanations.

Our statistical analysis shows that for both *Win%* and for *PercentileRank*, participants in 'RGA' did not perform statistically better than those in 'None', **invalidating H1a**, nor statistically better those in 'Hints', **invalidating H1b**. We do observe that 'RGA+' participants performed significantly better than 'None' participants **validating H1c**. However, with respect to the 'Hints' condition, 'RGA+' had statistically better *Win%* performance but not *PercentileRank*, therefore only **partially validating H1d**. Finally, we see no significant difference between 'RGA+' and 'RGA' conditions in this study, **invalidating H1e**.

In summary, these results indicate that humanly interpretable rationales *can* improve task performance, as long as the rationales fit a complete representation of the domain. Our results show that gathering both utility-based features and additional domain knowledge features (not represented by the utility) can accomplish completeness. It is also important to note that for task improvement, a decision not only needs to be interpretable, but more importantly *humanly* interpretable by accompanying rationales. The 'Hints' conditions also provided interpretability by highlighting the best move, but the lack of accompanying rationales may be the cause of why no significance was seen between 'Hints' and 'None'. Furthermore the inclusion of domain knowledge in RGA+ significantly improved participant *Win%* over the baseline conditions, compared to RGA, validating the need for a more complete domain representation. However, for this reason, it is interesting that no *Win%* significance was seen between 'RGA+' and 'RGA', but in seeing a trend of increasing difference between 'RGA+' and 'RGA', we suspect that over a longer period of time, a visible significance would be seen.

6.2 Participant Perceived Performance

Figure 7 presents the perceived performance rating (*SelfEval*) of participants in each experimental condition. The Likert scale data shows that participant groups that received more justifications ('RGA+', 'RGA') had higher ratings of 'Agree' and 'Strongly Agree' than participant groups that received little to no justifications ('Hints', 'None'), showing the value justifications had on *SelfEval*. Furthermore, participant groups that received some justifications ('RGA', 'Hints') had more 'Neutral' ratings than participant groups that were on the extreme ends of the justification spectrum ('RGA+', 'None'), showing higher levels of uncertainty in their *SelfEval*.

The Mann-Whitney U tests in Table 1 further detail the specific significance between each experimental condition. As seen in Table 1, the *SelfEval* of 'RGA' participants were not significantly stronger than 'None' participants' ratings, **invalidating H2a**, nor significantly stronger than 'Hint' participant ratings, **invalidating H2b**. However, the 'RGA+' participants did have a statistically higher *SelfEval* than 'None' participants and 'Hints' participants on all three days, **validating H2c** and **validating H2d** respectively.

Additionally, from day two onward, 'RGA+' participants rated their perceived performance higher than those from 'RGA', **validating H2e**.

Overall, the *SelfEval* metric data aligns with the performance analysis from Section 6.1, showing that perceived performance ratings are significantly stronger with the presence of humanly interpretable rationales that are domain representative. The results above also portray additional significance not seen with the *Win%* and *PercentileRank* metrics. Unlike the trend in Section 6.1, *SelfEval* does show a significant difference in performance rating between 'RGA+' and 'RGA', reiterating the importance of holistic domain representation. Interestingly, similar to the analysis from *Win%* and *PercentileRank*, *SelfEval* also does not portray a significant difference between 'RGA' and 'Hints', implying that rationales from the utility function alone were not different enough from 'Hints'. In Figure 7, we see an increasing difference in 'Disagree' ratings and 'Agree' ratings over Day 1 and Day 3 between 'RGA' and 'Hints', implying that over a longer period of observation, a potential significance could be seen.

7 DISCUSSION AND CONCLUSIONS

In this work, we are the first to explore whether human-interpretable rationales automatically generated based on an AI's internal task representation can be used to not just explain the AI's reasoning, but also enable end users to better understand the task itself, thereby leading to improved user task performance. Our work introduces the *Rationale-Generating Algorithm* that utilizes utility based computational methods to produce rationales understandable beyond the scope of domain-expert users. To validate RGA, we applied it to the domain of chess and measured human task performance using both qualitative user self-reported data (self-perceived performance ratings) and quantitative performance measures (winning percentages and rank percentiles of the strength of moves played by each participant).

Our results show that rationales from RGA are effective in improving performance when information from the AI's utility function is combined with additional domain knowledge from an expert. The resulting system was able to statistically significantly improve user performance in chess compared to study participants who practiced the same number of games but did not receive rationales. Simply showing participants the optimal action without an accompanying rationale did not produce the same results, indicating the importance of interpretable rationales in elucidating the task.

The presented approach is the first study of how rationales can affect learning. While it contributes a number of important insights, it is also limited in several ways. First, RGA is limited to utility-based methods and can not be applied to arbitrary machine learning methods. Future work in generating rationales for alternate ML representations, such as reinforcement learning discussed in [12], should be explored. Second, our work does not compare among the many different ways in which rationales can be phrased or structured. Additional research is needed to evaluate how to present rationales in the most accessible and interpretable manner, including how to tailor rationales to varying levels of expertise.

REFERENCES

[1] Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y Lim, and Mohan Kankanhalli. 2018. Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. ACM, 582.

[2] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.

[3] Michael Bain and Stephen Muggleton. 1994. Learning optimal chess strategies. In *Machine intelligence 13*. Oxford University Press, Inc., 291–309.

[4] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. 2017. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504* (2017).

[5] Jacky Bourgeois, Janet Van Der Linden, Gerd Kortuem, Blaine A Price, and Christopher Rimmer. 2014. Conversations with my washing machine: an in-the-wild study of demand shifting with self-generated energy. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 459–470.

[6] Kristy Elizabeth Boyer, Robert Phillips, Amy Ingram, Eun Young Ha, Michael Wallis, Mladen Vouk, and James Lester. 2011. Investigating the relationship between dialogue structure and tutoring effectiveness: a hidden Markov modeling approach. *International Journal of Artificial Intelligence in Education* 21, 1-2 (2011), 65–81.

[7] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1721–1730.

[8] Neil Charness, Michael Tuffiash, Ralf Krampe, Eyal Reingold, and Ekaterina Vasyukova. 2005. The role of deliberate practice in chess expertise. *Applied Cognitive Psychology* 19, 2 (2005), 151–165.

[9] Keith Cheverst, Hee Eon Byun, Dan Fitton, Corina Sas, Chris Kray, and Nicolas Villar. 2005. Exploring issues of user model transparency and proactive behaviour in an office environment control system. *User Modeling and User-Adapted Interaction* 15, 3-4 (2005), 235–273.

[10] Enrico Costanza, Joel E Fischer, James A Colley, Tom Rodden, Sarvapali D Ramchurn, and Nicholas R Jennings. 2014. Doing the laundry with agents: a field trial of a future smart energy system in the home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 813–822.

[11] Anind K Dey. 2018. Context-Aware Computing. In *Ubiquitous computing fundamentals*. Chapman and Hall/CRC, 335–366.

[12] Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark O Riedl. 2019. Automated rationale generation: a technique for explainable AI and its effects on human perceptions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. ACM, 263–274.

[13] Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*. 3429–3437.

[14] Maria Fox, Derek Long, and Daniele Magazzeni. 2017. Explainable planning. *arXiv preprint arXiv:1709.10256* (2017).

[15] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820* (2018).

[16] David Gunning. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web* 2 (2017).

[17] David Gunning and David W Aha. 2019. DARPA's Explainable Artificial Intelligence Program. *AI Magazine* 40, 2 (2019), 44–58.

[18] Mohanad M Hilles and Samy S Abu Naser. 2017. Knowledge-based Intelligent Tutoring System for Teaching Mongo Database. (2017).

[19] Feng-hsiung Hsu. 1999. IBM's deep blue chess grandmaster chips. *IEEE Micro* 19, 2 (1999), 70–81.

[20] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. 2018. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell* 172, 5 (2018), 1122–1131.

[21] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 1675–1684.

[22] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics* 9, 3 (2015), 1350–1371.

[23] Ali O Mahdi, Mohammed I Alhabbash, and Samy S Abu Naser. 2016. An intelligent tutoring system for teaching advanced topics in information security. (2016).

[24] Nick Montfort and Ian Bogost. 2009. *Racing the beam: The Atari video computer system*. Mit Press.

[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 1135–1144.

[26] Herbert Simon and William Chase. 1988. Skill in chess. In *Computer chess compendium*. Springer, 175–188.
[27] Stockfish. [n.d.]. http://stockfishchess.org/.
[28] Alistair G Sutcliffe, Frans van Assche, and David Benyon. 2016. *Domain knowledge for interactive system design*. Springer.
[29] Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. 2018. Multinet: Real-time joint semantic reasoning for autonomous driving. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1013–1020.
[30] Mike Wu, Michael C Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. 2018. Beyond sparsity: Tree regularization of deep models for interpretability. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[31] Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. 2018. Interpreting cnn knowledge via an explanatory graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[32] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. 2019. Interpreting cnns via decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6261–6270.
[33] Quan-shi Zhang and Song-Chun Zhu. 2018. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering* 19, 1 (2018), 27–39.