

XAlgo: Explaining the Internal States of Algorithms via Question Answering

Juan Rebanal
UCLA HCI Research
jrebanal@g.ucla.edu

Yuqi Tang
UCLA HCI Research
yukitang0703@gmail.com

Jordan Combitsis
UCLA HCI Research
jcombitsis17@gmail.com

Kai-Wei Chang
UCLA Computer Science
kwchang@cs.ucla.edu

Xiang ‘Anthony’ Chen
UCLA HCI Research
xac@ucla.edu

ABSTRACT

Algorithms often appear as ‘black boxes’ to non-expert users. While prior work focuses on explainable representations and expert-oriented exploration, we propose XAlgo—a generalizable interactive approach using question answering to explain deterministic algorithms to non-expert users who need to understand the algorithms’ internal states (*e.g.*, students learning algorithms, operators monitoring robots, admins troubleshooting network routing). We contribute a formal model that first classifies the type of question based on a taxonomy, and generates an answer based on a set of rules that extract information from representations of an algorithm’s internal states, *e.g.*, the pseudocode. A user study in an algorithm learning scenario with 18 participants (9 for a Wizard-of-Oz XAlgo and 9 as a control group) reports what kinds of questions people ask, how well XAlgo responds, and what remain as challenges to bridge users’ gulf of understanding algorithms.

Author Keywords

Explainable AI; algorithm; non-expert users; Wizard-of-Oz.

CCS Concepts

•**Human-centered computing** → **Human computer interaction (HCI)**; User studies;

INTRODUCTION

The world is increasingly run by automation algorithms [54], most of which function as ‘black boxes’ to people with limited algorithm expertise. As a result, non-expert users are often left with no ways to comprehend how or why an algorithm arrives at certain (unexpected) results, causing a lack of transparency and explainability in algorithm-driven scenarios.

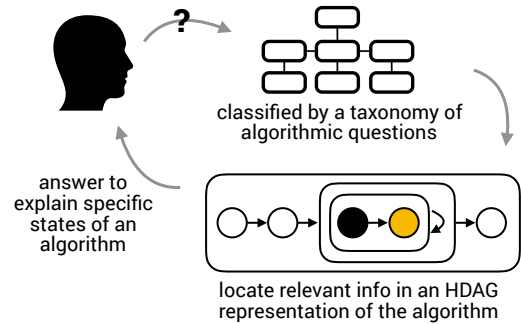


Figure 1. XAlgo is a generalizable interactive approach using question answering to explain deterministic algorithms to non-expert users. Our main contribution is a formal model that first classifies a user’s question by a taxonomy, which guides our proposed model to locate relevant algorithm states and extract relevant information to compose an answer.

To solve this problem, prior work has focused on explainable representations via extracting rules to approximate an algorithm’s behavior [15, 47, 49], visualizing components of an algorithm [40, 41], or tracing parts of the input to attribute the algorithm’s output [52, 70]. Others have gone beyond static representations to provide interactive tools that support visual analysis of algorithms for computing experts (*e.g.*, developers, data scientists) [57, 35, 29, 55, 26, 30], or to enable non-expert domain users (*e.g.*, physicians [65, 60]) to explore and understand the underlying computational model.

Most of the interactive algorithm explanations have been taking what Wick and Thompson [61] called a *reconstructive* approach: instead of ‘breaking into’ a black-box model, one can reconstruct a model’s behavior by visualizing, inspecting and tweaking its input and output data. The rationale is that by reconstructing a more explainable ‘proxy’ that approximates the original model, one can allow non-expert users to obtain a level of understanding beyond a black box.

However, a reconstructive approach presents limited insight for users who do need to understand the internal mechanisms of algorithms, *e.g.*, students learning sorting functions, a network administrator troubleshooting the system’s routing policy, and a remote operator analyzing what causes a field robot’s abnormal behavior. On the spectrum of expertise, such a group of people are situated between developers and end-

users: although they do not have the first-hand knowledge as the algorithm developers, they do need to achieve a level of algorithmic understanding that goes beyond end-users.

Our goal is to support these users not only by showing them how algorithms work, but, more importantly, to answer algorithm-related questions asked by users. Specifically, we focus on explaining deterministic algorithms that do not depend on statistical training data.

We propose XAlgo (eXplainable Algorithm)—a formal model that generates answers to a user’s question related to specific states of an algorithm’s execution. As shown in Figure 1, XAlgo consists of two steps: (i) identifying the type of questions—we draw from literature [36, 20, 27, 16] and more specifically NLP research on question answering [19, 67, 21, 32, 64] to build a taxonomy of questions one might ask (what, why, how and yes/no) in relation to understanding an algorithm; (ii) composing the answer—based on the identified question type, we propose a set of rules for extracting information from representations of an algorithm (e.g., the pseudocode), which serve as building blocks of composing the answer.

To ensure that we get “the right design” [58] as opposed to “a design right”, we employ a Wizard-of-Oz method to validate XAlgo: rather than testing users’ acceptance of a system design (e.g., [17, 14, 31]), we study the performance of XAlgo’s model (similar to [34, 7, 25])—i.e., whether the model can answer users’ questions to help them understand an algorithm.

Example Scenario: XAlgo for Learning Algorithms

We instantiate XAlgo in one of the aforementioned scenarios—a tutoring system for students to learn sorting functions—as a Wizard-of-Oz platform on top of VisuAlgo¹—a website that curates animations of data structures and algorithms. Figure 2 shows a snapshot of a QuickSort algorithm [22] where one of our study participants asked “Why did 8 and 2 move?”. XAlgo classifies this question as both Causality and Rationale related, and composes the following answer (annotations in brackets are added for illustration purposes):

*“Because 2 is less than the pivot, 3. [causality]
We swap 2 and 8 to build the subarrays. [local rationale]
This lets us sort the pivot, 3, into the correct position [global rationale]”*

The first sentence explains causality—that 2 being smaller than the pivot 3 causes the algorithm to swap 2 and 8. The second sentence explains the ‘local’ rationale—the immediate goal where sorting 2 and 8 is to separate elements into two subarrays: one smaller and the other greater than the pivot 3. The third sentence explains a ‘global’ rationale by going one level higher—by having the subarrays sorted we can then put the pivot at the correct position (between the subarrays).

Based on this algorithm learning scenario, we conducted a study with 18 participants² (9 for a Wizard-of-Oz XAlgo and 9 for a control group). Specifically, the wizard followed XAlgo’s

¹<https://visualgo.net/en>

²All with < 1 year programming experience and < 3 college-level programming courses.

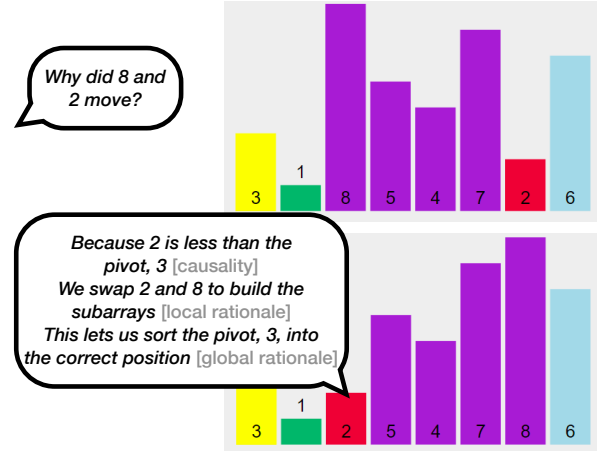


Figure 2. One of the many scenarios XAlgo can be useful for is learning algorithms, e.g., answering a user’s question about an animation of a sorting algorithm

model to manually generate answers on-the-fly to participants’ questions. Results show that (i) participants asked a wide variety of questions covering most parts of the taxonomy (Figure 3); (ii) participants found that XAlgo’s answers provided useful and accurate information that grounded their understanding of the algorithm, although formulating questions, algorithmic vocabulary and information processing remained the three main challenges; (iii) neither group’s participants performed well in quizzes that tested their learning of the algorithm, suggesting that for learning, more active guidance is needed beyond (passively) waiting to answer users’ questions.

We close by discussing the technical feasibility of implementing XAlgo, as well as design recommendations for future development of Explainable AI systems.

Contributions

Our main contributions are as follows:

- Conceptual contribution: an interactive question answering approach for explaining the internal states of deterministic algorithms;
- System design contribution: a formal model that first identifies the type of algorithm-related question and then composes answers based on existing representations of the algorithm, e.g., pseudocode;
- A Wizard-of-Oz study that validates the formal model and summarizes further design and implementation recommendations.

Limitations

We summarize our limitations here and provide a detailed discussion later in the paper.

- Our scope is in explaining deterministic algorithms rather than data-driven learning models;
- Rather than general, concept-related questions, our model is designed to answer questions related to specific execution states of an algorithm;

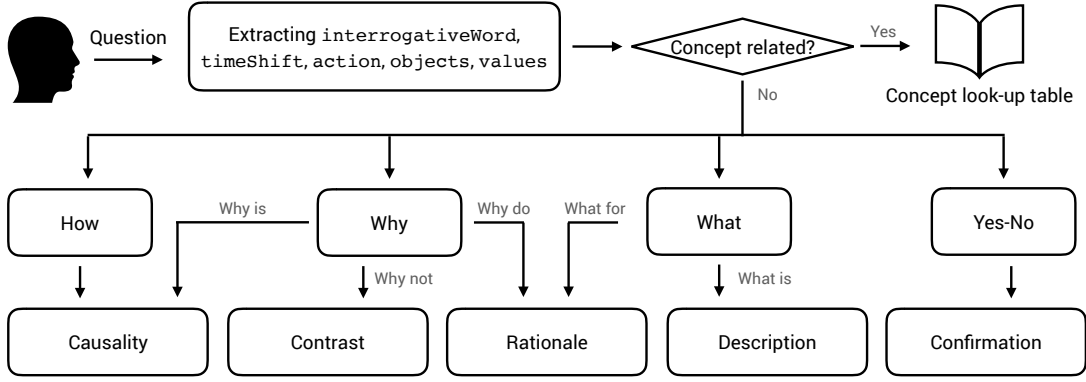


Figure 3. XAlgo classifies an algorithm-related question by first extracting key information and determining whether the question is asking about concept (that can be answer by looking up a table of pre-generated concepts); if not, the question is classified first by its interrogative word, and finally into one of the five question types, which are illustrated with detailed examples in Figure 4.

- In order to fully automate XAlgo, additional NLP components have to be developed (which we leave as future work), *e.g.*, to classify a why-question as Causality, Contrast or Rationale, to conduct semantic parsing to analyze the questions and to generate natural language answer based on predefined templates, and to collect and annotate a large corpus of algorithm-related natural language data;

FORMAL MODEL

XAlgo responds to an explanation-seeking question by first classifying the question into one of the five major categories (Figure 3), and then generating the answer using a set of rules to extract information from the relevant internal states of the algorithm (Figure 5). Below we detail the model’s process of classifying questions and generating answers.

Question Classification

Question classification is a well-established problem in NLP (*e.g.*, [8, 19, 21, 33, 32, 24, 67, 38]). As shown in Figure 3 and detailed in this subsection, the main difference of our model is the last ‘layer’ that follows the initial *wh*-word based classification, where we taxonomize five categories of explanation a question might be seeking.

As a user asks a question about the algorithm, the model first extracts the following information required for the subsequent classification of the question:

- *interrogativeWord*: also known as the *wh*-word, *e.g.*, why, what, how. Such words usually occur at the beginning of a question and are used to identify the overall question type;
- *timeShift*: whether the user is asking about past, now or future states of the algorithm based on words *e.g.*, next, last, after, as well as grammatical tense.
- *objects*: algorithm entities, *e.g.*, elements in an array;
- *values*: parameters of *objects*, *e.g.*, the number carried by each array element;
- *action*: what the algorithm does with the *objects*, *e.g.*, swapping two elements in an array;

Next, the model filters questions that are concept related. In the informal pilot studies, we noticed a subset of questions that were related to the concepts of the algorithm, rather than to the specific states of the ongoing algorithmic process. For example, “What is this algorithm trying to do?”, “What is a pivot?”, and “Why is that number selected as the pivot?”. Currently we answer such concept-related questions using a manually one-time generated look-up table, although more advanced, automatic methods can be used (*e.g.*, based on mining code comments [63]), which we leave as future work.

If the question is not concept-related, the model then classifies the question based on its interrogative word as one of the five question types as shown in Figure 4.

Answer Generation

Based on the identified type of question, the next step is to locate relevant internal states of an algorithm and to extract relevant information from that state to generate an answer. We first introduce a general representation XAlgo uses to model and traverse an algorithm’s states.

Hierarchical Directed Acyclic Graph. Pseudocode is a commonly used representation for algorithms, as it concisely and universally describes the logic of the algorithm independent of the algorithm’s implementation or the top-level application. To create a more convenient way to look up information in the pseudocode, we employ a data structure akin to the Hierarchical Directed Acyclic Graph (HDAG) [56]. Essentially, an HDAG transforms pseudocode into a state diagram with hierarchy. The rules of creating an HDAG are as follows:

- To create hierarchies: an algorithm starts as a (root) DAG, and each loop, conditional branch or recursive call creates a child DAG. In this way, a DAG can be thought of as a high-level operation, which is manifested as branches or iterations of executions under its hierarchy; conversely, a low-level operation (*e.g.*, a node) is being governed by or is contributing to the parent DAG’s operation.
- To create nodes: if a line of pseudocode causes a value change of objects in the main data structures, we create a new node (however, other changes such as iterator *i++* does

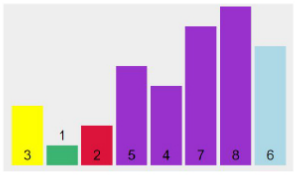
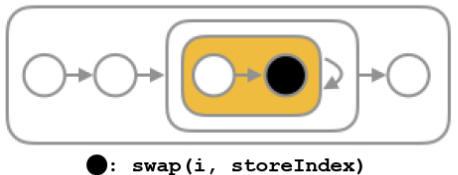
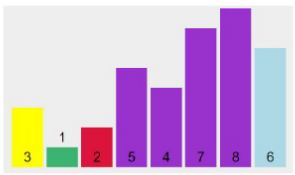
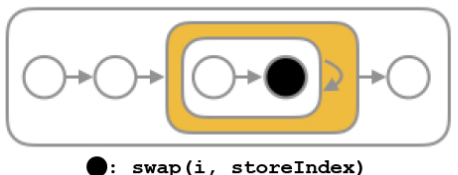
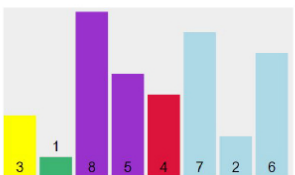
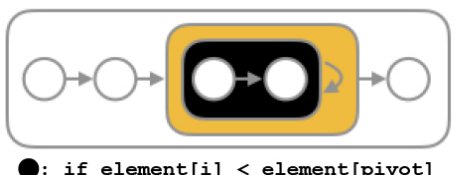

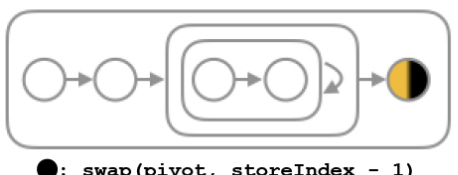
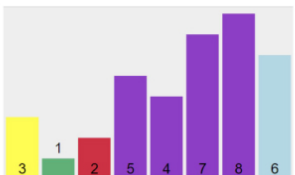
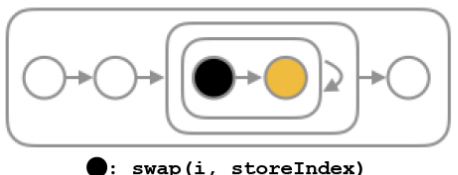
Question type	Example question	Generation process	Final answer
Causality Past actions that caused the current values of certain objects	 <p>"Why did 2 and 8 move?"</p>	 <p>●: <code>swap(i, storeIndex)</code></p> <ul style="list-style-type: none"> The parent node is conditional (if), thus used as the answer node Related information: answer node's conditional logic 	<p>"Because the current element, 2, is smaller than the pivot, 3"</p> <p>↑ the conditional (if) logic</p> <p>"We swap the current element, 2, with the third element, 8."</p> <p>↑ the current node's action</p>
Rationale Reasons for certain actions on certain objects	 <p>"Why did 2 and 8 move?"</p>	 <p>●: <code>swap(i, storeIndex)</code></p> <ul style="list-style-type: none"> The parent node has no goal; use the grandparent node as the answer node; Related information: answer node's goal 	<p>"We swap the current element, 2, with the third element, 8"</p> <p>↑ the current node's action</p> <p>"So we can sort the current partition into a subarray smaller than the pivot, 3, and a subarray larger than the pivot, 3"</p> <p>↑ the grandparent node's goal</p>
Contrast Identifying differences between comparable situations	 <p>"Why didn't 5 move when compared with the pivot?"</p>	 <p>●: <code>if element[i] < element[pivot]</code></p> <ul style="list-style-type: none"> Use the parent node as answer node Key information: answer node's goal 	<p>"If 5 moves, we will not sort the current partition into a subarray smaller than the pivot, 3, and a subarray larger than the pivot, 3"</p> <p>↑ negate the parent node's goal</p>
Description ... of the action that takes place on certain objects	 <p>"What just happened?"</p>	 <p>●: <code>swap(pivot, storeIndex - 1)</code></p> <ul style="list-style-type: none"> The current node is the answer node Key information: answer node's action 	<p>"We swapped the pivot, 3, with the third element, 1"</p> <p>↑ the current node's action</p>
Confirmation ... of a hypothesis related to future actions, objects, and values.	 <p>"Is storeIndex incremented after swap?"</p>	 <p>●: <code>swap(i, storeIndex)</code></p> <ul style="list-style-type: none"> The nearest node with swap action is the current node "After" indicates the answer node is the subsequent node In the answer node: "storeIndex" refers to the object storeIndex, "increment" is synonymous to the action ++ 	<p>"Yes"</p> <p>↑ information matches between question and the answer node</p>

Figure 4. Examples of answer generation for each question type classified from Figure 3 based on the VisuAlgo animation of a QuickSort algorithm.

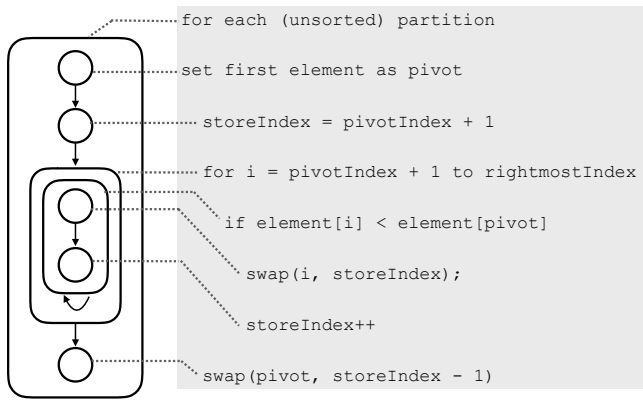


Figure 5. An exemplar Hierarchical Directed Acyclic Graph (HDAG) constructed from the pseudocode of the QuickSort algorithm: each loop or conditional block creates a new hierarchy whereas each line of code can be represented as a node that contains objects (e.g., an array), values (e.g., numbers stored in an array), action (e.g., swapping two array elements), and goal (a description of what this node does).

not warrant new nodes). The idea is that each node corresponds to a perceivable intermediate state of an algorithm, e.g., array elements changing positions while being sorted. In this way, a question asking about the algorithm at the current state can be mapped to a specific node for answer generation.

Figure 5 shows an exemplar HDAG representation of the QuickSort algorithm on a simple array for illustration purposes. Importantly, besides the basic algorithmic constructs (*object*, *value*, *action*), each node should also contain a description of its *goal*, which can be annotated as a one-time preprocessing step, similar to how programmers comment their code. For example, the for-loop in QuickSort (Figure 5) has a goal of “sorting the pivot in the right place”, and the if-statement has a goal of “compare the current element and the pivot”.

Locating the answer node Based on the HDAG representation, we first identify the current node associated with the current state which the question is asked about. Specifically, we use *timeShift*—if there is any—to ‘shift’ to the antecedent or subsequent nodes. For example, as shown in Figure 4, the question “Is storeIndex incremented after swap?” causes a shift to the subsequent node.

We then continue to locate the ‘answer node’—the actual node where the answer can be generated. If the question is Description or Confirmation, the answer node is the current node. Figure 4 shows several specific examples. If it is a Causality question, the model considers two cases: if the node is nested in a conditional statement, we use the node that contains that statement as the answer node; otherwise we use the current node’s antecedent as the answer node. For example, as shown in Figure 4, “Why did 2 and 8 move?” finds the answer node as its parent node, which is an if-statement (see corresponding source code in Figure 5). If it is a Rationale or Contrast question, we use the current node’s parent as answer node. For example, as shown in Figure 4, the same question

“Why did 2 and 8 move” can also be answered with a rationale, which is the goal of its grandparent node.

It is possible that a user might ask a ‘context-independent’ question, i.e., the answer to which is unrelated to the current node of algorithm execution. In such cases, we find the nearest node that matches a user’s question.

Extracting information to compose an answer. Once the answer node is located, we employ the following rules to generate answers. If a question is Causality, Rationale or Description, we simply describe the answer node’s *goal* (with some natural language components to address the specific question type, e.g., ‘because’, ‘so’, ‘if’). If a question is Contrast, we negate the answer node’s *goal*. If a question is Confirmation, we compare the *objects*, *values* and *action* between the question and the answer node, and inform the user whether their hypothesis matches the referred state in the algorithm.

The last column in Figure 4 shows examples of generated answers. Note that within the scope of this paper, all the natural language components were manually added, which can be further automated as a post-processing step. We leave this as future work.

USER STUDY

To validate XAlgo’s model without the cost of system implementation, we conducted a Wizard-of-Oz study of XAlgo in an algorithm learning application scenario. Specifically, we sought to answer the following research questions:

- RQ1. What kinds of questions do users ask XAlgo?
- RQ2. How well can XAlgo answer users’ algorithm-related questions?
- RQ3. How well can users achieve the application-specific task, i.e., learning an algorithm, using XAlgo?

To investigate these questions, we situate XAlgo in an algorithm learning/understanding scenario based on the aforementioned VisuAlgo platform (Figure 6). We chose QuickSort as the target of explanation, as it is a commonly used algorithm. Compared to ‘entry-level’ sorting algorithms (e.g., Bubble Sort), QuickSort has a certain amount of complexity that could elicit explanation-seeking questions from users trying to learn the algorithm.

Design & Participants

We employed a between-subject design. The independent variable was *System* (XAlgo’s QA+animation vs. VisuAlgo’s animation-only). We recruited 18 participants—9 for each condition—via convenience sampling, flyers in a local university and a Craigslist online posting. We employed a screening questionnaire as people signed up for the study. Specifically, we only accepted participants with none or less than one year of programming experience, who had taken fewer than three college-level computer science courses. Specific to the algorithm chosen for the study, we also made sure participants had no or limited understanding of sorting (by asking them to describe QuickSort in the screening questionnaire). Amongst the 18 participants, there were 8 male, 10 female, aged 19 to 25.

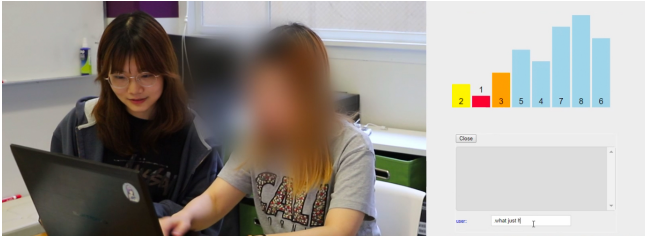


Figure 6. We conducted a Wizard-of-Oz study of XAlgo deployed on the VisuAlgo platform. The second experimenter (the wizard) was in the same room but not shown in this image.

None of the participants had studied or was studying any fields related to Computer Science or Electrical Engineering. Participants were randomly assigned to one of the two conditions. Each participant received a \$25 gift card as compensation.

Apparatus & Setup

To implement XAlgo’s model, we used VisuAlgo’s pseudocode to pre-populate an HDAG with one-time annotations for answer generation. To implement XAlgo as a test platform, we modified the VisuAlgo codebase by adding a custom dialog box (Figure 6). This allowed for the use of VisuAlgo’s visualizations and pseudocode to naturally prompt the user to ask questions about the algorithm. We also added a communication protocol so that two XAlgo programs can talk to each other over a local area network, thus enabling the wizard’s performance of XAlgo to a user. For the VisuAlgo group, we ran the unmodified VisuAlgo codebase locally.

The experiment took place in the conference room of our research lab. One experimenter guided the participant; for the XAlgo group, another experimenter played the wizard, who was in the same room using a different laptop to chat with the participants. XAlgo participants were told that the wizard was taking notes and to provide technical support when needed.

Each participant’s screen was recorded during the entire study using Open Broadcaster Software. Meanwhile, audio was recorded using the laptop’s microphone.

Tasks & Procedure

Introduction & training (5 min). We started by introducing the common background and motivation, which was to provide an interactive environment for people to learn and understand algorithms. For the XAlgo group, we explicitly instructed participants to ask questions in order to understand the algorithm. Next, we provided a hands-on tutorial of the system (VisuAlgo or XAlgo) and let each participant try out the interaction.

Interacting with the system & multiple-choice quizzes (30 min). Once participants familiarized themselves with the interaction, they were asked to interact with the system, trying to learn and understand QuickSort by watching how an array was sorted (VisuAlgo) and also by asking questions (XAlgo). Participants were given six multiple-choice quiz questions they must finish before the session ends.

In the XAlgo group, upon receiving a participant’s question, the wizard instantly responded with “Processing ...” and began generating an answer on the fly following the aforementioned model. In the VisuAlgo group, both experimenters did not answer any of participants’ algorithm-related questions.

In the six multiple choice questions, a participant was given a subsequence of VisuAlgo animation (of a different array) and asked to predict what was the next step.

We capped this phase to 30 minutes, although participants could choose to end early.

Walkthrough quizzes (10 min) & interviews (15 min). In the next phase, participants were asked to walk through a complete process of sorting a simple four-element array by filling in the blanks of the key steps and describing how the array was updated along the way. Participants were given a maximum of 10 minutes to finish the quiz.

Then, we conducted a semi-structured interview where the participants evaluated the difficulty of the learning task, how much more they felt they understood the algorithm after interacting with VisuAlgo/XAlgo, and whether they felt the learning process was enjoyable.

For the XAlgo group, we further asked participants about whether they could understand the generated answers, whether the answers were natural, detailed and accurate, and whether the answers helped them learn and understand the algorithm. We also asked participants to comment on other positive aspects of the overall experience and areas for improvement.

FINDINGS

Summary of Findings

We provide a summary of our findings before discussing the details.

1. Participants asked a wide variety of questions covering most parts of the taxonomy: nearly half the questions were concept-related that did not directly contribute to their understanding of the algorithm;
2. Participants found that XAlgo’s answers provided useful and accurate information that grounded their understanding of the algorithm, although formulating questions, algorithmic vocabulary and information processing remained the three major challenges;
3. Neither group’s participants performed well in quizzes that tested their learning of the algorithm, suggesting that for learning, more active guidance is needed beyond (passively) waiting to answer users’ questions.

#1 What questions non-computing users asked XAlgo

During the study, the wizard followed the aforementioned classification model (Figure 4) to label the question type. After the study, the other experimenter reviewed each participant’s question and, without seeing the wizard’s label, perform another question type classification. Then the two experimenters resolved the differences via discussion. One change we made was merging the categories of Rationale and Causality questions, as we found in the informal pilot study that combining

answers to both types of questions provided a more comprehensive explanation that covers both ‘how come’ and ‘what for’ [37]. Figure 4 shows such an example.

Overall, XAlgo participants asked a total of 92 questions. Four were excluded as they were not algorithm-related (*e.g.*, “What should I do?”), leaving 88 questions answered by XAlgo—almost an average 10 questions per person. To our surprise, amongst these questions 37 (42.0%) were concept-related, which was answered using a pre-generated look-up table (although more advanced, automatic methods are possible, which we leave as future work). The look-up table encompassed all the conceptual topics (*e.g.*, the concept of a pivot) but did not anticipate all the possible questions. For example, while the table contained an answer to “What is a pivot?”, it did not provide a direct response to questions such as “Does the pivot always have to be number in the middle?”; in such cases, the wizard would simply answer the question with the closest information from the table.

Our focus is on the other 51 (58.0%) questions where 21 (23.9%) were Causality/Rationale, 17 (19.3%) were Confirmation, 12 were Description (13.6%) and only one was Contrast. The high concentration of concept-related questions was likely because participants were unfamiliar with algorithmic terms. Unfortunately, learning these terms would not be enough to understanding an algorithm; rather, it should be the XAlgo-type of ‘operational’ questions that actually reveal an algorithm’s process to a user. Further, it was a little surprising that only one question was Contrast—the reason might be participants were still at the novice phase and were not able (or not confident) to develop a contrastive hypothesis to ‘challenge’ the algorithm.

We summarize the recurring themes to outline key factors that contribute to XAlgo’s ability to answer users’ algorithm-related questions, as well as challenges for future work.

#2 How well XAlgo answers users’ algorithm questions

Figure 7 summarizes participants’ responses that evaluated the answers generated by XAlgo (referred to as ‘chatbot’ in the questions).

To obtain further insight behind these ratings, we employed an iterative open-coding method to analyze the qualitative data collected from the semi-structured interview that accompanied these questions. Two experimenters coded each participant’s data one day after the study. One experimenter performed the first pass of coding and updated a codebook, which was then reviewed by the other experimenter to discuss and resolve disagreements. The two experimenters alternated the roles of first coder and reviewer. After all participants’ data was coded and consolidated, a third experimenter reviewed all the codes and transcripts, and resolved disagreements through discussion with the previous two experimenters.

The main recurring themes are as follows.

Groundedness Asking question externalizes the internal obstacle a person experiences in understanding a concept. One main benefit of XAlgo was helping participants ground their

1: Strongly disagree — 7: Strongly agree									
1	2	3	4	5	6	7	Mean	Median	SD
The language the chatbot uses is natural									
-	1	2	1	-	3	2	4.89	6.00	1.90
The chatbot can understand my question well									
-	-	-	1	3	1	4	5.89	6.00	1.17
The chatbot’s reply is helpful for understanding the algorithm									
-	-	-	2	2	3	2	5.56	6.00	1.13
The chatbot’s reply is detailed enough									
-	-	-	-	-	4	5	6.56	7.00	0.53
The chatbot’s reply is accurate enough									
-	-	-	-	3	2	4	6.11	6.00	0.93

Figure 7. Participants’ responses to XAlgo’s answers. Overall participants reacted positively to XAlgo’s ability to answer their algorithm questions: the most agreed-upon aspect is providing sufficient details whereas naturalness has the most split ratings.

understanding of an algorithm to a specific state that gave rise to a question:

... sometimes I just like assume I know something while I don’t; so this like explains very well (P5)

Formulating questions Despite the usefulness of grounding via questions, a number of participants expressed difficulty in coming up with what questions to ask in the first place or formulating what they wanted to ask into a question.

I don’t really know how to word my questions like I asked it and then once it answers if I’m still confused, maybe you can like ask further questions. (P5)

Participants suggested that XAlgo could guide a user to ask questions, or how to ask questions more strategically, *e.g.*, “advise people to ask for definitions first” (P1).

Design implications Develop a database of questions so that the system can suggest questions to a user either as a ‘warm-up’ exercise, or to assist them when they struggle to formulate their own question.

Vocabulary, at the beginning of the session, was a main obstacle for participants to fully comprehend XAlgo’s answers.

... they confuse me just because I think, the vocab[ul]ary I didn’t totally understand yet; but now that I do understand the vocab it’s a lot easier to understand. (P1)

In this study, XAlgo generated answers from the algorithm’s pseudocode and comments provided by VisuAlgo, and used the majority of the terms as they are, *e.g.*, subarray, partition, and pivot. We did prepare a concept table in case participants asked about these algorithmic terms; yet we observed few participants would ask additional questions just to clarify terms used in the previous answer.

Design implications To help users overcome the initial gulf of vocabulary, future design can hyperlink algorithmic terms to their definitions, or more proactively explain these terms when the user first encounters them, or, as P1 pointed out, even encourage participants to ask about definitions first.

Information processing Participants pointed out that XAlgo’s answer was “*too wordy*” (P7) and “*gives details but too much details*” (P2). This problem stemmed from a design choice—we traded off conciseness with completeness to ensure that the answer includes all the necessary details.

Design implications To mitigate users’ information processing load, some participants suggested a better structure, *e.g.*, “*break it up into more sentences*” (P4). Perhaps another solution—and a greater challenge—is to deliver a lengthy answer conversationally, allowing users to ask follow-up questions that gradually reveal ‘the complete picture’.

Directness Related to information processing, participants tended to favor direct answers whenever possible, *e.g.*, a yes/no to a Confirmation question.

... gave me like clear-cut answers and gave me like a yes and no one. And I wanted a yes or no ... (P7)

For the other types of answers, participants felt that useful and necessary information was provided but not in a direct manner, *e.g.*, “*in a roundabout way*” (P6):

... it does like give you useful answers even though if they’re not like the ones that directly answered the question you asked. (P6)

Design implications Indeed, XAlgo’s answer generation focuses on retrieving question-relevant information. However, there is a gap between having the requisite information and delivering such information in a way that is directed to the user’s question. Future work can approach this challenge by learning from empirical examples (similar to [12]) or using data-driven generative methods [10].

Naturalness was mentioned a few times amongst participants: while they understood the utility of the answer to provide useful and accurate information, the unnaturalness was also fairly noticeable (as shown in the wide distribution of ratings in Figure 7):

... it answers everything accurately and it gives the information that I asked for but it does so like sounding more like a glossary like a dictionary ... (P8)

... just use the vocabulary that you need anyway, so it’s not going to like, you know, sound like a conversation ... (P4)

... like a robot’s answers ... If I asked someone to explain it, it wouldn’t give me all this. (P2)

Design implications The pursuit of a highly human-like natural language response has been a long-standing challenge that is yet to be fully realized in the future. As a compromise, people do develop ‘unnatural’ ways of ‘speaking’ to a system as they become familiar expert users. Thus a pragmatic solution is for XAlgo to define a language that can jointly achieve the goal of explanation while being learnable by a human user, even if it means not answering like a human does.

1: Strongly disagree — 7: Strongly agree											Mann-Whitney	
	1	2	3	4	5	6	7	Mean	Median	SD	U	p
I think the learning task is easy												
Control	-	3	3	2	1	-	-	3.11	3	1.05	21.00	0.094
XAlgo	-	1	2	1	5	-	-	4.11	5	1.17		
I understand the algorithm after interacting with the system												
Control	-	1	1	2	3	2	-	4.44	5	1.33	24.00	0.16
XAlgo	-	1	1	-	2	1	4	5.44	6	1.88		
I think the learning process is enjoyable												
Control	1	1	1	-	2	2	2	4.67	5	2.18	27.00	0.26
XAlgo	-	-	-	2	-	4	3	5.89	6	1.17		

Figure 8. Participants’ response to the overall learning experience: in all three questions, XAlgo scored at least one point higher; the differences, however, were not statistically significant.

#3 How well participants achieved the learning tasks

Both XAlgo and the control group participants made progress but still struggled in understanding the algorithm.

Quizzes results

One experimenter graded participants’ quizzes based on pre-generated keys. We conducted Mann-Whitney U tests, the results of which show that there is no significant difference between XAlgo and the control group in either multiple-choice scores ($U = 38.0$, $p = 0.863$) or sorting walkthrough scores ($U = 40.0$, $p = 1.000$). The overall average combining the two groups is 4.56/6 for multiple choice and 5.83/10 for sorting walkthrough.

In hindsight, the quiz questions overall might have been too difficult for a user who had just started to learn an advanced sorting algorithm less than 30 minutes before. However, the more fundamental challenge, we believe, is that to achieve an advanced level of understanding an algorithm, a user would need *active* guidance. XAlgo’s approach, on the other hand, remains passive. As discussed in the previous subsection, oft-times participants did not know what questions to ask that would advance their understanding of an algorithm. Perhaps a future design of XAlgo should be able to proactively prompt and guide users to ask meaningful questions, or even generate questions to quiz users as a way to provide active guidance.

Learning experience: understanding, difficulty & enjoyment

We asked participants from both XAlgo and the control group to rate and comment on their overall learning experience via interacting with the system. Figure 8 summarizes the quantitative results. Overall, XAlgo’s ratings on easiness, understanding and enjoyment were at least one point higher; the differences, however, were not statistically significant.

To further obtain qualitative insight, we conducted a brief semi-structured interview asking each participant to elaborate their rating on the three questions. We made two observations that illustrated the difference of participants’ experience of XAlgo compared to the control condition.

1. We noticed that only the control group participants expressed ‘surprise’:

I thought I understood the algorithm, but I don’t think I did fully. (P12)

... you think you got it and then it's actually not right (P13)

XAlgo's ability to answer participants' questions allowed them to verify issues that they were unsure about, whereas in the control group participants had no way of asking questions and had to make their own assumptions, at times causing 'surprises' when they found out that their assumptions were actually wrong.

2. The control group participants wished there was explanations of the VisuAlgo animation.

[I would have liked] more explanation on the code—I don't know what's what (P10)

[If] I had more guidance alongside with them and I think [I] would be able to figure it out. And probably more time. But someone like explaining it to me would be better. (P12)

RELATED WORK

To inform the design of a question-answering approach for explaining algorithms, we first review two key related bodies of work: explainable AI and question-answering research. We further review prior work on understanding software—a similar objective but a different audience (developers); and past research on intelligent tutoring systems—a similar approach (learning) to achieve the understanding of algorithm.

Explainable AI (XAI)

Foremost, our work is motivated by a lack of explainability in 'black boxes' AI-enabled systems. A large body of recent work has focused on making data-driven AI models explainable, as the representations of such models (*e.g.*, a neural network) often do not permit a user to understand how a model works, why it works, or why it does not work. Ras *et al.* summarize three families of XAI methods [46]: (i) *rule-based*, where rules are extracted that match how a 'black-box' model process certain input to produce certain output (*e.g.*, [15, 48, 49]); (ii) *attribution*, where certain output is attributed to specific parts of the input or specific components of the model (*e.g.*, [53, 40, 9, 41, 43]); and (iii) *intrinsic methods*, elements of a model that are intrinsically explainable (*e.g.*, [69]) without adding any rules or attribution. Doran *et al.* review AI-enabled systems with a spectrum of explainability, ranging from opaque, to interpretable, to comprehensible, and to (truly) explainable [11]. Hoffman *et al.* discuss metrics to evaluate XAI, including the goodness of explanations, whether users are satisfied by explanations, how well users understand the AI systems, how curiosity motivates the search for explanations, whether the user's trust and reliance on the AI are appropriate, and how the human-XAI work system performs [23]. Miller conducted a comprehensive review on XAI in connection with relevant disciplines, *e.g.*, communication, cognitive science and sociology [37].

The goal of our paper differs the prior work above: rather than explaining data-driven AI (*e.g.*, neural networks), we focus on process-driven algorithms: whereas the former approach is based on a statistical model parameterized by training data, we are interested in the latter, where an algorithm

itself self-sufficiently describes the logic of solving a problem. Although algorithms already have a representation of the problem-solving process, such representations (*e.g.*, pseudocode, block diagram, finite state machine) remain obscure to people without a computational background. Wilhelme *et al.* propose a shape analysis technique that visualizes key states and structural properties of a data structure (*e.g.*, a heap) during the execution of an algorithm [62]. Shakshuki *et al.* develop SHALEX—a system for explaining algorithms that addresses a lack of multi-level abstraction and user interaction in prior algorithm visualization [28]; a follow-up paper further provides users with the ability to write their own explanations for events as a way of learning [51]. One important observation related to all this work, as pointed out by Miller, is that explanation is by nature social—"a transfer of knowledge, presented as part of a conversation or interaction" [37]. Below we switch our attention to one of the most relevant form of conversation that can potentially facilitate explanation-question answering.

Question Answering (QA)

Question answering (QA) intersects natural language processing and information retrieval. The process of achieving QA can be decomposed into two steps: interpreting a question and generating an answer. Below we sample a selected amount of prior work to illustrate the background of QA research.

Classifying a question involves associating a question with a category or a topic where there is sufficient knowledge base for answer generation. Mohasseb *et al.* take a grammar-based approach to classify a question into one of the six categories: confirmation, factoid, choice, hypothetical, causal, and list [38]. However, such a static categorical label is topic-invariant and does not provide insight of a question's actual content, *i.e.*, which topic the question-asker is interested in. To classify the topic of a question, prior work has taken a language-centric approach, *i.e.*, following rules and models of human language to parse the question. Hermjakob *et al.* develop 'QTargets'—a type of question-answering classes, and employed a hierarchical tree-like model for parsing a question based on specific components of the speech [21]. Li *et al.* take a language modeling approach and classify questions by determining whether certain questions belong to the same dialogue [32]. Girju *et al.* focus on the extraction of causal relations in a question by keywords and develop three types of 'causatives': simple, resultative and instrumental [16]. In the meantime, complementary to using a language model, researchers also take a data-driven approach for question classification. Others perform coarse-to-fine classification of the topic associated to a question [32, 24]. Beyond question classification, there is another school of NLP work focused on semantic parsing to a specific structure or format [5, 4, 66].

Generating an answer is often considered as an information extraction (IE) problem, which has already accumulated a large body of prior research. For example, Yih *et al.* consider 'single-relation question', *e.g.*, "When were DVD players invented?"; the answer is an entity that can be connected to another entity in the question ('DVD players') by a relation (be-invented-in) [68]. Nie *et al.* augment community QA, using textual answers provided by an online community to gather rel-

evant multimedia information on the web to enrich the answer [39]. The advent of data-driven learning gives rise to new QA development based on large corpora of examples [45, 44]. Our research on XAlgo analogously augments existing animated illustration of algorithms by extracting information at each step to answer a user’s context-dependent questions. Indeed, the use of visuals is often inseparable from QA: Visual QA uses a neural model to recognize relevant entities in an image and uses that information to compose an answer [3]. Specifically, we are interested in extracting answers to algorithm-related questions, which intersects two major application domains: support for understanding software and tutoring technology based on chatbots, which we review below.

Understanding Software Codes & Programs

Algorithm explanation is often manifested as source code exploration and understanding in software engineering. For example, Alber *et al.* employ an attribution-like method to explain an image classification program by attributing a result to specific parts of a software project [1]. Oney *et al.* further incorporate communication as a means to facilitate explanation of source code, providing integrated support for instructors and learners to chat with one another with directed references to the source code [42]. To generate explanation-specific answers about a program’s behavior, Ko *et al.* retrieve a causal chain—a flow of actions that lead the specific outcome questioned by the user [2]. Wood *et al.* collected a corpora of QA conversations during bug repair and distill a set of “speech acts” that signal specific tasks in question [64]. Different the scope of our research, all this work focuses on professional software developers; for users without a computational background, being explained about an algorithm is almost indistinguishable from learning that algorithm. Below we briefly review intelligent tutoring systems, focusing on systems that employ chatbots.

Intelligent Tutoring Systems using Chatbots

The use of chatbot-like natural language based interaction is a widely adopted approach in building intelligent tutoring systems. AutoTutor is a dialogue-based tutor that provides conversation-facilitating features, *e.g.*, giving different good/bad/neutral feedback, prompts, “pumps” (inviting the learner to speak up), and hints to promote an interactive learning experience [18]. Van Lehn *et al.* find that question-answering serves as an effective means to fill the granular ‘gaps’ between the student and the tutor [59]. Fei *et al.* find that chatbots as a communicative approach to learning allows learners to study in low-anxiety situations and allow learners to engage material when they feel ready: students are more willing to repeat themselves/ask chatbots to repeat themselves [13]. Ruan *et al.* find that students strongly prefer using a chatbot to learn and memorize factual knowledge compared to traditional flash cards [50]. Cai *et al.* develop a rule-based chatbot for learning math and attain a strong preference when employed in an online video tutorial system [6].

LIMITATIONS, DISCUSSIONS AND FUTURE WORK

Practicality of Implementing XAlgo

We took a Wizard-of-Oz approach to validate how well the formal model of XAlgo works should it be implemented. In order

for future work to actually implement XAlgo, one major task is to realize the requisite NLP capabilities. Specifically, question type classification can be implemented based on classification techniques such as SVM or logistic regression; answer generation can be implemented based on either hand-crafted templates specific to each explanation type or by taking a data-driven generative approach (similar to [12]). Perhaps the main challenge is generating conceptual or state-related annotations. There are two solutions for future work: (i) promoting and enabling a best practice for developers to create such annotations as they author an algorithm, which can be supported via a tool that extracts structures from static analysis of the code and prompts developers to enter comments for specific states; (ii) for algorithmic components that are commonly used (*e.g.*, sorting routines), future work can take a data-driven approach to mine large corpora of publicly available source code (*e.g.*, [63]) and aggregate developers’ comments as material for XAlgo’s answer generation.

Extending XAlgo for Data-Driven Statistical Algorithms

Although data-driven, statistical algorithms (*e.g.*, neural networks) can also be thought of as a network of states, the fundamental difference is that such states (*e.g.*, neurons in a neural network) are too low-level to represent high-level logic. Thus it would be futile to naïvely apply XAlgo to locate, *e.g.*, the excitation of a specific neuron. Rather, we expect XAlgo to be combined with existing research on high-level visualization of a neural network, *e.g.*, Olah *et al.*’s approach that can indicate patterns of object recognition at various levels (*e.g.*, neuron, channel, layer) [40]. In particular, XAlgo can consider these visualization building blocks the same way it treats different states in an HDAG and provide an interactive mechanism for users to navigate such visualizations.

Using XAlgo to Provide Interactive, User-Centered XAI

XAlgo represents an underexplored XAI approach where the explainable medium is no longer static representation of an algorithm, but an interactive dialog that allows users to ask for specific kinds of explanations they deem useful. In this way, a system can provide a user-centered explanation, rather than explanations derivative of an algorithm’s existing representations. Specifically, XAlgo works best for explaining a specific class of AI-enabled systems where states are constantly changing as a response to environmental or user input. For example, consider GPS navigation. Using XAlgo, a driver can ask “Why not taking the highway?”, which is classified as a Contrast question. Through the navigation algorithm’s internal logic, XAlgo identifies that choosing local streets vs. highway is to achieve an immediate goal of avoiding traffic. Thus XAlgo answers “If taking the highway, you will not avoid traffic.”

Exploring More Algorithms and More Scenarios

XAlgo can be used for deterministic algorithms that behave like a finite state machine. We focus on one specific example—QuickSort—for illustrating the mechanism of XAlgo and for enabling a specific application scenario. One promising future work is to develop XAlgo into a toolkit that developer can use for explaining their own programs, which also allows for testing XAlgo’s model on a larger variety of algorithms.

XAlgo is generalizable and application-agnostic. While future work should validate such generalizability in wider range of scenarios, in this paper we chose to focus on one specific application—learning a sorting algorithm—so that we can perform an in-depth study to gather empirical evidence on how well XAlgo works: specifically, what kinds of questions people asked, whether XAlgo provides value-added responses, and what remain as challenges to bridge users’ gulf of understanding algorithms.

REFERENCES

1. Maximilian Alber. 2019. Software and application patterns for explanation methods. (2019). <http://arxiv.org/abs/1904.04734>
2. J Ko Andrew and A Myers Brad. 2004. Designing the whyline: a debugging interface for asking questions about program behavior. *Proceedings of the SIGCHI conference on Human factors in computing systems* 6, 1 (2004), 151–158.
3. Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*. DOI: <http://dx.doi.org/10.1109/ICCV.2015.279>
4. Yoav Artzi and Luke Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics* 1 (2013), 49–62. DOI: http://dx.doi.org/10.1162/tac1_a_00209
5. Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, 1533–1544. <https://www.aclweb.org/anthology/D13-1160>
6. William Cai, Joshua Grossman, Zhiyuan Lin, Hao Sheng, Johnny Tian, Zheng Wei, Joseph Jay Williams, and Sharad Goel. 2019. MathBot: A Personalized Conversational Agent for Learning Math. (2019). DOI: http://dx.doi.org/10.475/123_4
7. Minsuk Chang, Anh Truong, Oliver Wang, Maneesh Agrawala, and Juho Kim. 2019. How to Design Voice Based Navigation for How-To Videos. (2019), 1–11. DOI: <http://dx.doi.org/10.1145/3290605.3300931>
8. William Chisholm, Louis T Milic, and John AC Greppin. 1984. *Interrogativity: A colloquium on the grammar, typology and pragmatics of questions in seven diverse languages, Cleveland, Ohio, October 5th 1981-May 3rd 1982*. Vol. 4. John Benjamins Publishing.
9. Piotr Dabkowski and Yarin Gal. 2017. Real Time Image Saliency for Black Box Classifiers. Nips (2017). <http://arxiv.org/abs/1705.07857>
10. Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision*. 2970–2979.
11. Derek Doran, Sarah Schulz, and Tarek R. Besold. 2018. What does explainable AI really mean? A new conceptualization of perspectives. In *CEUR Workshop Proceedings*.
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 *International Conference on Intelligent User Interfaces, Proceedings IUI*. DOI: <http://dx.doi.org/10.1145/3301275.3302316>
13. Yi Fei and Stephen Petrina. 2013. Using Learning Analytics to Understand the Design of an Intelligent Language Tutor – Chatbot Lucy. *International Journal of Advanced Computer Science and Applications* (2013). DOI: <http://dx.doi.org/10.14569/ijacsa.2013.041117>
14. Juliana Jansen Ferreira, Ana Fucs, and Vinícius Segura. 2019. Should I Interfere? AI-Assistants’ Interaction with Knowledge Workers. *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems - CHI ’19* (2019), 1–7. DOI: <http://dx.doi.org/10.1145/3290607.3299052>
15. Nicholas Frosst and Geoffrey Hinton. 2018. Distilling a neural network into a soft decision tree. In *CEUR Workshop Proceedings*, Vol. 2071. <http://arxiv.org/abs/1711.09784>
16. Roxana Girju. 2003. Automatic detection of causal relations for Question Answering. (2003), 76–83. DOI: <http://dx.doi.org/10.3115/1119312.1119322>
17. John D Gould, John Conti, Todd Hovanyecz, Todd Hovanyecz, and Madison Ave. 1983. Composing Letters with a Simulated Listening Typewriter. April (1983).
18. Arthur C. Graesser, Shulan Lu, George Tanner Jackson, Heather Hite Mitchell, Mathew Ventura, Andrew Olney, and Max M. Louwerse. 2004. AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, and Computers* 36, 2 (2004), 180–192. DOI: <http://dx.doi.org/10.3758/BF03195563>
19. Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*. 219–224.
20. Robert James Hankinson. 2001. *Cause and explanation in ancient Greek thought*. Oxford University Press.
21. Ulf Hermjakob and Ulf Hermjakob. 2000. Parsing and Question Classification for Question Answering. (2000).
22. Charles AR Hoare. 1962. Quicksort. *Comput. J.* 5, 1 (1962), 10–16.

23. Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for Explainable AI: Challenges and Prospects. (2018), 1–50. <http://arxiv.org/abs/1812.04608>
24. Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *EMNLP 2008 - 2008 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference: A Meeting of SIGDAT, a Special Interest Group of the ACL*.
25. Dhruv Jain, Angela Lin, Rose Guttman, Marcus Amalachandran, Aileen Zeng, Leah Findlater, and Jon Froehlich. 2019. Exploring Sound Awareness in the Home for People who are Deaf or Hard of Hearing. (2019), 1–13. DOI: <http://dx.doi.org/10.1145/3290605.3300324>
26. Minsuk Kahng, Dezhi Fang, and Duen Horng Polo Chau. 2016. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM, 1.
27. Alex Kass and David Leake. 1987. Types of explanation. (1987), 1–75. <https://apps.dtic.mil/docs/citations/ADA183253>
28. Andreas Kerren. 2006. for Improving Algorithm Teaching. 1, 212 (2006), 175–176.
29. Josua Krause, Adam Perer, and Kenney Ng. 2016. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 5686–5697.
30. Josua Walter Hugo Krause. 2018. *Using Visual Analytics to Explain Black-Box Machine Learning*. Ph.D. Dissertation. New York University Tandon School of Engineering.
31. Bokyoung Lee, Sindy Wu, Maria Jose Reyes, and Daniel Saakes. 2019. The Effects of Interruption Timings on Autonomous Height-Adjustable Desks that Respond to Task Changes. (2019), 1–10. DOI: <http://dx.doi.org/10.1145/3290605.3300558>
32. W. Li. 2002. Question classification using language modeling. *CiteSeer* (2002). <http://scholar.google.com/scholar?hl=en>
33. Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1–7.
34. David Maulsby, David Maulsby, Saul Greenberg, Richard Mander, and Mariani Ave. 1993. Wizard of Oz Prototyping an Intelligent Agent through Wizard of Oz. (1993).
35. Thorsten May, Andreas Bannach, James Davey, Tobias Ruppert, and Jörn Kohlhammer. 2011. Guiding feature subset selection with an interactive visualization. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*. IEEE, 111–120.
36. Tim Miller. 2018. Contrastive Explanation: A Structural-Model Approach. (2018), 1–30. <http://arxiv.org/abs/1811.03163>
37. Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019), 1–38. DOI: <http://dx.doi.org/10.1016/j.artint.2018.07.007>
38. Alaa Mohasseb, Mohamed Bader-El-Den, and Mihaela Cocea. 2018. Question categorization and classification using grammar based approach. *Information Processing and Management* (2018). DOI: <http://dx.doi.org/10.1016/j.ipm.2018.05.001>
39. Liqiang Nie, Xiaochi Wei, Dongxiang Zhang, Xiang Wang, Zhipeng Gao, and Yi Yang. 2017. Data-driven answer selection in community QA systems. *IEEE transactions on knowledge and data engineering* 29, 6 (2017), 1186–1198.
40. Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature Visualization. *Distill* 2, 11 (nov 2017), e7. DOI: <http://dx.doi.org/10.23915/distill.00007>
41. Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. The Building Blocks of Interpretability. *Distill* 3, 3 (mar 2018), e10. DOI: <http://dx.doi.org/10.23915/distill.00010>
42. Steve Oney, Christopher Brooks, and Paul Resnick. 2018. Creating Guided Code Explanations with chat.codes. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–20. DOI: <http://dx.doi.org/10.1145/3274400>
43. Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2018. Multimodal Explanations: Justifying Decisions and Pointing to the Evidence. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. DOI: <http://dx.doi.org/10.1109/CVPR.2018.00915>
44. Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for SQuAD. *arXiv preprint arXiv:1806.03822* (2018).
45. Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).
46. Gabrielle Ras, Marcel van Gerven, and Pim Haselager. 2018. Explanation methods in deep learning: Users, values, concerns and challenges. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*. Springer, 19–36.
47. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. 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.

48. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (feb 2016), 1135–1144. <http://arxiv.org/abs/1602.04938>
49. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. *Thirty-Second AAAI Conference on Artificial Intelligence* (apr 2018). <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewPaper/16982>
50. Sherry Ruan, Liwei Jiang, Justin Xu, Bryce Joe-Kun Tham, Zhengneng Qiu, Yeshuang Zhu, Elizabeth L Murnane, Emma Brunskill, and James A Landay. 2019. QuizBot: A Dialogue-based Adaptive Learning System for Factual Knowledge Sherry. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19 Chi* (2019), 1–13. DOI: <http://dx.doi.org/10.1145/3290605.3300587>
51. Elhadi Shakshuki, Andreas Kerren, and Tomasz Müldner. 2007. Web-based Structured Hypermedia Algorithm Explanation system. *International Journal of Web Information Systems* 3, 3 (2007), 179–197. DOI: <http://dx.doi.org/10.1108/17440080710834238>
52. Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17)*. JMLR.org, Sydney, NSW, Australia, 3145–3153.
53. Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. (dec 2013). <http://arxiv.org/abs/1312.6034>
54. Christopher. Steiner. 2012. *Automate this : how algorithms came to rule our world*. Portfolio/Penguin. 248 pages. <https://www.amazon.com/Automate-This-Algorithms-Came-World/dp/B00D9T9IQG>
55. Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, Alexander M Rush, and others. 2016. Visual analysis of hidden state dynamics in recurrent neural networks. *arXiv preprint arXiv:1606.07461* (2016).
56. Jun Suzuki, Tsutomu Hirao, Yutaka Sasaki, and Eisaku Maeda. 2003. Hierarchical directed acyclic graph kernel: Methods for structured natural language data. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 32–39.
57. Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S Tan. 2009. EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1283–1292.
58. Maryam Tohidi, William Buxton, Ronald Baecker, and Abigail Sellen. 2006. Getting the right design and the design right. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 1243–1252.
59. Kurt VanLehn. 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist* 46, 4 (2011), 197–221. DOI: <http://dx.doi.org/10.1080/00461520.2011.611369>
60. Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing theory-driven user-centric explainable AI. In *Conference on Human Factors in Computing Systems - Proceedings*. DOI: <http://dx.doi.org/10.1145/3290605.3300831>
61. Michael R. Wick and William B. Thompson. 1992. Reconstructive expert system explanation. *Artificial Intelligence* (1992). DOI: [http://dx.doi.org/10.1016/0004-3702\(92\)90087-E](http://dx.doi.org/10.1016/0004-3702(92)90087-E)
62. Reinhard Wilhelm, Tomasz Müldner, and Raimund Seidel. 2002. Algorithm explanation: Visualizing abstract states and invariants. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.
63. Edmund Wong, Taiyue Liu, and Lin Tan. 2015. Clocom: Mining existing source code for automatic comment generation. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 380–389.
64. Andrew Wood and Paige Rodeghero. Detecting Speech Act Types in Developer estion / Answer Conversations During Bug Repair. (????). DOI: <http://dx.doi.org/10.475/123>
65. Yao Xie, Melody Chen, David Kao, Ge Gao, and Xiang 'Anthony' Chen. 2020. CheXplain: Enabling Physicians to Explore and Understand Data-Driven, AI-Enabled Medical Imaging Analysis. In *To appear at the 2020 CHI Conference on Human Factors in Computing Systems*.
66. Xiaojun Xu, Chang Liu, and Dawn Song. 2018. SQLNet: Generating structured queries from natural language without reinforcement learning. (2018).
67. Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014a. Semantic Parsing for Single-Relation Question Answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, 643–648. DOI: <http://dx.doi.org/10.3115/v1/P14-2105>
68. Wen Tau Yih, Xiaodong He, and Christopher Meek. 2014b. Semantic parsing for single-relation question answering. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*.
69. Qiao Zheng, Hervé Delingette, and Nicholas Ayache. 2019. Explainable cardiac pathology classification on

cine MRI with motion characterization by semi-supervised learning of apparent flow. *Medical Image Analysis* (2019). DOI: <http://dx.doi.org/10.1016/j.media.2019.06.001>

70. Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. 2017. Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. (Feb. 2017).