Brainstormer: A Model of Advice-Taking*

Eric K. Jones
Institute for the Learning Sciences
Northwestern University

Abstract

Research on advice-taking in artificial intelligence is motivated by the promise of knowledge-based systems that can accept high-level, human-like instruction [11]. Examining the activity of human advice-taking is a way of determining the key computational problems that a fully-automated advice taker must solve.

In this paper, we identify three features of human advice-taking that pose computational problems, and address them in the context of BRAINSTORMER, a planning system that takes advice in the domain of terrorist crisis management.

1 The Advice-Taking Task

Advice is information communicated with the intent of helping an agent make progress on a problem. Taking advice therefore involves understanding how the advice is relevant to solving a problem and acting on this understanding. We have identified three characteristic features of the human advice-taking task, each of which imposes important computational demands on the advice taker. We believe that a fully-automated advice taker has to give an account of each of these features.

1. **Empowerment:** Taking advice empowers an agent to do something by furnishing him or her with information. Consequently, an advice taker must be capable of translating initial, high-level advice into instructions that can be carried out.
2. **Situatedness:** Advice addresses the needs of an agent situated in the context of trying to solve a particular problem. It follows that an advice taker should be able to relate advice to needs for information that have arisen in the course of problem solving.
3. **Communication:** People never communicate everything that is relevant; instead, they provide only information sufficient for a recipient to reconstruct the intended message from context. Advice is communicated information; an advice taker thus has to be able to infer information that is relevant but not explicitly provided.

In the remainder of this section, we describe the computational demands that each of these features of advice-taking place on an advice taker, and discuss the extent to which past research has addressed these demands.

1.1 Advice-taking as empowerment

Most past research on advice-taking has focused on showing how taking advice can empower an agent to do something. Research into this aspect of advice-taking falls into two categories. One category of research has investigated the problem of converting high-level, human-like advice into concrete or *operational* instructions that a machine can carry out [6, 11, 12]. This research advocates that view of an advice taker as a kind of compiler for a very high-level language. Mostow's program FOO [12], for example, translates high-level advice about playing the game of hearts, (*"avoid taking points"*), into a heuristic-search problem solver that can carry out that advice.

*This research was supported in part by the Air Force Office of Scientific Research (AFOSR). The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization.

A second category of research starts with the observation that that a major difficulty in empowering knowledge-based systems is the so-called knowledge acquisition bottleneck: the problem of getting knowledge into the system sufficient for it to perform interesting tasks. This leads naturally to the view of an advice taker as mechanism for streamlining knowledge acquisition [1, 2, 3, 14]. The TEIRESIAS system [1, 2], for example, guides knowledge acquisition in a way that encourages completeness and consistency with existing knowledge, and also provides high-level tools to facilitate debugging problems that arise from faulty or missing knowledge.

1.2 Advice-taking as a situated activity

Unfortunately, more is involved in taking advice than converting it into an operational vocabulary and ensuring consistency. Once one considers advice-taking as a *situated* activity, further demands are placed on the advice taker. In particular, a situated advice taker must be able to use the advice it receives to address needs for information that arise in the context of current problem solving.

Systems such as FOO and TEIRESIAS do not treat advice-taking as a situated activity. In FOO, for example, there is no current problem solving because no problem solver exists prior to advice-taking. Instead, FOO builds a complete problem solver from scratch during advice-taking, using the input advice as a specification. TEIRESIAS is concerned with acquiring knowledge for unspecified future problem solving, not acquiring knowledge needed to progress on a current problem.

There are systems that direct advice towards current problem-solving, ([5, 10, 16] for example), but they all make assumptions that drastically simplify the interaction between advice giver and advice taker: they assume that advice is supplied immediately after a need for information arises, and that the advice comes with detailed instructions about how it is to be used. For example, the Soar system [5] can use search-control advice in current problem solving, but advice is provided as soon as an operator selection difficulty is encountered, and the system simply presumes that any advice it receives is intended to be used to resolve that particular difficulty.

In a more realistic setting, neither of these simplifying assumptions holds. First, an advice-giver cannot be expected to be on hand the instant a need for information arises. A problem solver that is engaged in a complex problem solving task can be expected to have a large set of current needs, some of which it may succeed eventually in satisfying by itself, others of which may require outside assistance. An advice giver should be able to give advice that relates to any current need, whether or not it is in the current "focus of attention" of the problem solver.

Second, in a real-world setting, an advice giver cannot be expected to give detailed instructions about how advice is to be used. An ability to give such instructions requires that the advice giver know the exact nature of the problem solver's needs, which in turn requires that it have a detailed knowledge of the arcane internal workings of the problem solver. For example, the instructible Soar system described above requires that the user know about operator selection and about the details of particular operators, both of which kinds of knowledge should be hidden from the user.

For these reasons, a fully automated advice taker must be equipped to relate advice to the problem solver's current needs for information. This implies that the advice taker know what these needs are. At the very least, the advice taker should be able to record needs for information as they arise, and store them in such a way that they can be accessed when advice is presented that addresses them.

1.3 Advice-taking as communication

Advice is not just information relevant to a specific problem, it is *communicated* information. An advice taker is guided by the expectation that the advice giver intends the advice to be helpful in current problem solving. For the advice to be helpful, it should address a current need for information. If there is an operational form of the advice that can easily be seen to satisfy a current need, then the advice taker's task is simple. But there will also arise situations where adducing the advice giver's intentions is less straightforward.

People expect discrepancies between the information that advice ostensibly provides and the information that they actually need. These discrepancies can arise in the course of human communication for reasons of communicative efficiency, or because the advice giver has an incorrect or incomplete understanding of the needs of the advice taker. In either case, the advice taker expects to have to bridge the

gap between an initial, "literal" interpretation of the advice and a derived interpretation that makes clear how the advice addresses an actual need for information.

Furthermore, even if a need for information can be determined that the advice plausibly addresses, the advice may not be presented in sufficient detail for it to adequately satisfy the need. In that case, the advice taker should try to fill in the missing details; failing that, it should ask the advice giver for more information.

While some existing advice-taking systems are capable of asking the user to fill in missing details [2], few are capable of inferring these details on their own, (the MOLE system [3] is a noteworthy exception). Furthermore, existing systems cannot bridge the gap between ostensible and actual needs for information. In the remainder of this paper, we introduce the BRAINSTORMER system, and show how considering advice-taking as situated activity that involves communication has informed BRAINSTORMER's design.

2 Brainstormer: A Planner That Takes Advice

Our theory of advice-taking is implemented in BRAINSTORMER, a planning system that takes advice in the domain of terrorist crisis management. BRAINSTORMER translates abstract advice into specific directives about how to make progress on a problem it is working on. Advice is presented in the form of proverbs, represented in an abstract vocabulary of planning-relevant knowledge[7, 13, 15]. BRAINSTORMER uses the advice to come up with plans for countering terrorism.

BRAINSTORMER has two top-level modules: the **planner** and the **adapter**. The planner is BRAINSTORMER's problem solver and is similar in some ways to STRIPS [4]. It suggests plans for countering terrorism. The adapter is BRAINSTORMER's advice taker. It acts as an intelligent assistant to the planner, using advice it receives to address pre-existing needs of the planner for information. In principle, the planner could operate on its own, but in practice it relies heavily on the adapter for assistance.

BRAINSTORMER provides an account of each of the aspects of advice-taking discussed above.

1. **Advice-taking as empowerment.** Like other advice takers, BRAINSTORMER's adapter has rules for translating initial, high-level advice into the operational vocabulary of the planner. Once this advice has been transformed into an operational vocabulary, it is in a form that can directly match the planner's needs for information. If suitable needs can be found, then the advice empowers the planner to make forward progress in planning.

2. **Advice-taking as a situated activity.** BRAINSTORMER is presented with advice during ongoing problem solving. The planner is first handed a planning problem and allowed to run to quiescence. Only then is advice presented. Although the planner come up with a variety of plan suggestions on its own, it will also have discovered needs for information that it is unable to satisfy by itself, and which if satisfied would allow it to make further progress in planning. These needs arise whenever the planner tries to retrieve a knowledge structure and fails. BRAINSTORMER monitors needs for information as they arise, recording them in memory as *queries*.

   Queries are stored in terms of features of potential answers. Once advice has been converted into the operational vocabulary of the planner, features of the advice are used as cues for retrieving queries that the advice might address. Attached to each query is a procedure that specifies how the planner should use answers to it. If a query is retrieved, and if the advice used to retrieve it constitutes an adequate answer, then the query's procedure is invoked. Planning can then continue at the point in the planning process where the need for information originally arose.

3. **Advice-taking as communication.** Taking advice in BRAINSTORMER, then, is defined as turning a proverb into an adequate answer to a pre-existing query. Proverbs are a very minimalist form of communication in which all details of the specific problem that they are intended to address are omitted. BRAINSTORMER thus expects input advice to be sketchy and incomplete. After translating advice into an operational vocabulary, there may still be no query

PROBLEM SITUATION: *A terrorist crisis involving terrorists from the P.L.O., in which five American hostages were taken and one was killed.*

GOALS: *Prevent future terrorist crises similar to the problem situation,* and *take action against the agents responsible for the terrorist crisis.*

Figure 1: An example planning problem.

1. *Find a plan for the goal of preventing future hostage holdings similar to the one in the problem situation*
2. *Find a plan for the goal of preventing future killings similar to the one in the problem situation*
3. *Find a plan for the goal of preventing future terrorist crises similar to the one in the problem situation*
4. *Find an explanation for the hostage holding*
5. *Find an explanation for the killing*
6. *Find an explanation for the terrorist crisis*

Figure 2: Some queries posted during planning.

in memory that directly matches it, or if there is, the advice may supply insufficient information to serve as an adequate answer.

It follows that BRAINSTORMER must confront the following problems: (1) finding queries that advice addresses and (2) turning advice into adequate answers. The adapter is equipped with two sets of inference strategies for coping with these problems. These strategies add elements to the advice that were not explicitly present in the input but which the advice giver may have intended to communicate implicitly. The first set of strategies allows the adapter to hypothesize additional components of the advice sufficient to retrieve a query that the advice addresses. Once a query is found, the second set of strategies allows the adapter to fill in missing details needed for the advice to provide an adequate answer to the query. In carrying out these kind of inferences, the adapter operates on the assumption that the user intended the input advice to be helpful but omitted certain details for the usual reasons that information is often left implicit in ordinary communication.

3 An Example Planning Problem and Initial Planning

In this section, we sketch a typical planning problem in BRAINSTORMER and describe some of the queries that arise from planning for this problem, prior to the presentation of advice. Planning problems are represented using **problem situations** and **goals**. A natural-language paraphrase of a representative planning problem is shown in figure 1, above. The plans that BRAINSTORMER comes up with are plans for goals in this problem.

BRAINSTORMER's planner is first invoked on this planning problem without the help of any advice and permitted to run to quiescence. In the process of planning, numerous queries are posted, each of which signals a potential opportunity for further planning. Some of these queries are listed in figure 2, above. Queries 1, 2, and 3 were posted when the planner failed to directly retrieve a plan for a goal. Queries 4, 5, and 6 request knowledge structures that can match preconditions of an abstract counter-planning rule. Given an explanation for the occurrence of an undesirable event, this rule can transform a goal to prevent the recurrence of a similar event into a goal to prevent the recurrence of the conditions that led to it[1]. Queries 4, 5 and 6 were posted on separate occasions where the planner tried to run this rule and failed to retrieve suitable explanations.

When the planner can make no further progress on its own, a user presents BRAINSTORMER with advice in the form of a proverb. BRAINSTORMER's

[1] In the context of goals to prevent future occurrences of similar events, an event is considered similar to a second event if there is an explanation for the first, a non-vacuous generalization of which is also an explanation for the second.

INPUT:	• The planning problem and queries described above. • The proverb *he who has suffered more than is fitting will do more than is lawful.*
OUTPUT:	• (Elaboration of the problem situation) *The terrorists are Palestinian, and were suffering because of poor living conditions in refugee camps.* • (Plan suggestion) *Build public housing projects for the refugees.*

Figure 3: An example that illustrates the problem of producing adequate answers

adapter then has the task of turning this proverb into an adequate answer to a query. In the previous section, we noted that in carrying out this task, the adapter must confront the problems of finding a query that the advice addresses and of fleshing out the advice into an adequate answer to it. As intimated above, these problems follow directly from the demands imposed by considering advice-taking as a situated activity involving communication. In the next two sections, we sketch BRAINSTORMER's approach to dealing with these problems.

4 Producing Adequate Answers

We start with the problem of adequate answers. Once a query has been found that advice addresses, BRAINSTORMER's adapter must fill in any details needed for the advice to constitute an adequately specific and plausible answer to the query. An apt illustration of this kind of processing arises during BRAINSTORMER's processing of the proverb *"he who has suffered more than is fitting will do more than is lawful."* BRAINSTORMER's input-output behavior is summarized in figure 3.

BRAINSTORMER came up with this output by using the proverb to answer query 6 above, (*"find an explanation for the terrorist crisis"*). One way that the proverb cen be represented in the operational vocabulary of the planner is as *an explanation for the occurrence of an illegal action: the illegal action happened because some agent was suffering, and that explains why the agent carried out the illegal action.* This interpretation directly matches queries 4, 5, and 6 above: hostage-holdings, killings, and terrorist crises can all be viewed as illegal actions, and the proverb can therefore provide an explanation for each. BRAINSTORMER prefers queries whose answers subsume answers to other queries; in this case, an explanation for the terrorist crisis subsumes explanations for the killing and hostage holdings, because each of these actions was part of the terrorist crisis. Consequently, BRAINSTORMER assumes that the advice is best treated as answering query 6[2].

Although the adapter possesses an operational representation of the advice and a query that it plausibly addresses, its task is not over. The advice is still missing important details which it is the adapter's responsibility to fill in. Once the adapter has committed to answering query 6, the representation of the advice can be paraphrased as follows: *the terrorists were suffering, and that explains why they carried out the terrorist crisis.* As it stands, this representation constitutes a hypothesis, that the terrorists were suffering, and that this explains why they carried out the terrorist crisis. For BRAINSTORMER to accept this hypothesis as an adequate answer to query 6, it requires that any unsupported components of the hypothesis be justified.

The adapter has a small set of strategies for justifying unsupported components of hypotheses. These include the following.

- **Memory search:** Use an unsupported component of a hypothesis as an index into memory for retrieving a knowledge structure that supports it.
- **Communicative assertion:** If only one unsupported component of a hypothesis remains, assume that the user intended the advice to assert that this component in fact holds.
- **Ask the user.**

Asking the user is hardly a novel idea: a variety of existing systems treat advice-taking as mixed-initiative knowledge acquisition, in which the system directs a user to fill in all missing details. In BRAIN-

[2]Our discussion here omits many important issues, including how the proverb is initially represented, how it is transformed into the operational vocabulary of the planner, and how the operational interpretation is used to retrieve relevant queries. For a discussion of these and other issues, see [7, 8, 9].

INPUT:	• The planning problem and queries described in section 2. • The proverb *an old poacher makes the best keeper.*
OUTPUT:	• (Plan suggestion) *Hire a former terrorist to defend against terrorism*

Figure 4: An example that illustrates the problem of finding relevant queries.

PRECONDITIONS:	**plan:** *a plan of defense* **parameter:** *the actor of the plan* **situation:** *the plan of defense is part of a goal conflict involving stereotyped attackers and defenders*
POSTCONDITION:	*A new plan whose actor satisfies the attacker's stereotype*

Figure 5: The proverb *"an old poacher makes the best keeper"* represented as a plan refinement operator.

STORMER however, asking the user is a strategy of last resort. Wherever possible, the adapter attempts to arrive at an adequate plausible interpretation of the advice on its own.

In the current example, there are two unsupported hypothesis components: that *the terrorists were suffering*, and that *this suffering explains why they carried out the terrorist crisis*. The adapter employs its **memory search** strategy to justify the first. Specifically, it uses the fact that the terrorists are members of the P.L.O. to retrieve a standard explanation for Palestinian suffering: poor living conditions in the refugee camps. The **communicative assertion** strategy is then sufficient to justify the second component.

Once the adapter has come up with what it judges to be an adequate answer to a query, the planner is reinvoked at the point in its processing that the query arose. Recall that this query was posted when the planner was trying to satisfy a precondition of a general counterplanning rule. The rule uses explanations of the occurrence of an undesirable event to transform a goal to prevent the recurrence of a similar event into goals to prevent the recurrence of the conditions that led to it. In the current instance, the undesirable event was the terrorist crisis. Failure to retrieve a suitable explanation from memory led the planner to post the query. The proverb provides the missing explanation: the terrorist crisis happened because of poor living conditions in the refugee camps. Now that it has this explanation, the planner can apply the counterplanning rule. This leads the planner to establish a new goal of improving living conditions in the refugee camps and eventually, to propose building public housing projects.

5 Finding Relevant Queries

In the preceding example, BRAINSTORMER encountered little difficulty trying to find a relevant query, because the operational form of the advice, (an explanation), directly matched a pre-existing query. In other cases however, this lucky state of affairs does not obtain. For example, suppose BRAINSTORMER is handed the proverb *"an old poacher makes the best keeper"*. BRAINSTORMER's input-output behavior is summarized in figure 4.

This time, the relevant operational form of the proverb is a *plan refinement operator*, which is a rule that the planner can use to elaborate plans it proposes. The representation of this operator is paraphrased in figure 5. This operator does not directly match any existing queries. If a suitable query already existed, the planner would have to have already considered a plan of defense for one of its goals, and be looking for ways to refine it. This however is not the case in the current example. Consequently, further work is necessary to find the query that the advice addresses.

The following is a highly informal sketch of the reasoning that the adapter goes through in searching for a relevant query. (For a more technical discussion, see [8]).

- The user has suggested a way of refining a plan of defense. She is giving me advice, so she is trying to help me solve my current problem, namely preventing future terrorist crisis similar to the one in the problem situation, (see figure 1). Thus, she believes that this plan refinement operator supplies information that is useful for this purpose. For this operator to be useful, a

plan of defense would also have to be useful. But I haven't considered any plans of defense. Maybe I should have considered a plan of defense for one of my goals.

- One of my goals is preventing future terrorist crises similar to the one in the problem situation. Could a plan of defense be useful in planning for this goal? Yes, it could. A plan of defense can be effective at preventing future occurrences of the kinds of events it defends against, because it raises their expected cost.
- OK, I'll *hypothesize* that a plan of this sort is actually useful for this goal in the current context. After all, maybe the user intended to communicate this hypothesis. That would explain why she believes that I should find this operator useful. Now I know how to use the operator.

In carrying out the above reasoning, the adapter locates a query that the advice addresses by elaborating the representation of the advice to include a hypothesis that a plan of defense is appropriate for the goal of preventing future terrorist crises. Next, control returns to the planner, which goes on to apply the plan refinement operator to this hypothesized plan and eventually suggests *hire a former terrorist to defend against terrorism.*

There are two important morals to draw from this example. First, the inference that the defense plan might be appropriate was *licensed by* the assumption that the user was trying to be helpful in the context of the given problem. This assumption both legitimates the final interpretation of the advice and justifies the inferential effort expended in finding it. An advice-taking system that does not reflect the idea that taking advice involves making sense of *information communicated with the intention of being helpful* is unable to make these inferences; consequently, in this situation at least, such an advice-taking system would be unable to profit from the advice.

The second moral of this example is that an advice taker needs knowledge about the problem solver's problem solving *process* in order to carry out this kind of inference. The requisite knowledge is over and above information about the advice-taker's dynamic needs for information. In the above example, the adapter had to be able to reason about how the planner *uses* plan refinement operators: it uses them when it has already considered a plan of defense for one of its goals. Carrying out this reasoning requires that the adapter know that the planner's planning process takes goals as input, suggests plans for those goals, and then suggests refinements to those plans. Previous advice-taking systems lack this kind of self-knowledge, and are consequently unable to perform this kind of reasoning.

6 Conclusion

We began this paper by identifying three characteristic features of human advice taking that should inform a computational theory of advice-taking. Advice *empowers* an advice taker, advice-taking is *situated* in the context of ongoing problem solving, and advice-taking involves *communication*.

BRAINSTORMER is an advice-taking system that takes account of these features of human advice-taking. First, advice helps BRAINSTORMER's planner to come up with plans for countering terrorism. Second, the planner's needs for information are recorded during planning as they arise, allowing BRAINSTORMER to use advice to facilitate ongoing problem solving. Third, BRAINSTORMER is equipped with inference strategies that allow it to infer some of the kinds of details that are commonly left implicit in ordinary human discourse. In summary, BRAINSTORMER represents a more comprehensive model of human advice-taking than earlier advice-taking systems, and constitutes a step in the direction of problem solvers that can accept high-level, human like instruction.

Acknowledgements

Thanks to Eric Domeshek for comments on a draft of this paper, and to Andy Fano for help with the program.

References

[1] Randall Davis. Knowledge acquisition in rule-based systems — knowledge about representation as a basis for system construction and maintenance. In D.A. Waterman and F. Hayes-Roth, editors, *Pattern Directed Inference Systems*. Academic Press, New York, 1978.

[2] Randall Davis. TEIRESIAS: Applications of meta-level knowlege. In Randall Davis and D.B. Lenat, editors, *Knowledge-Based Systems in Artificial Intelligence*, pages 229–484. McGraw-Hill, New York, 1982.

[3] Larry Eshelman and J. McDermott. MOLE: A knowledge acquisition tool that uses its head. In *Proceedings AAAI-86 Fifth National Conference on Artificial Intelligence*, pages 950–955, Philadelphia, PA., August 1986. AAAI.

[4] Richard E. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[5] Andrew Golding, P.S. Rosenbloom, and J.E. Laird. Learning general search control from outside guidance. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987. IJCAI.

[6] Frederick Hayes-Roth, P. Klahr, and D.J. Mostow. Advice-taking and knowledge refinement: An iterative view of skill acquisition. In John R. Anderson, editor, *Cognitive Skills and Their Acquisition*, pages 231–253. Lawrence Erlbaum Associates, Hillsdale, N.J., 1981.

[7] Eric K. Jones. Case-based analogical reasoning using proverbs. In Kristian Hammond, editor, *Proceedings: Case-Based Reasoning Workshop*, Pensacola Beach, FL., May 1989. Defense Advanced Research Projects Agency, Morgan Kaufmann Publishers, Inc.

[8] Eric K. Jones. Learning by taking advice: The need for a model of the problem-solving process. Paper submitted to the Seventh International Conference on Machine Learning, 1990.

[9] Eric K. Jones. BRAINSTORMER: A situated advice taker. Paper submitted to the Eighth National Conference on Artificial Intelligence, 1990.

[10] Richard L. Lewis, A. Newell, and T.A. Polk. Towards a SOAR theory of taking instructions for immediate reasoning tasks. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 514–521, Ann Arbor, Michigan, August 1989. Cognitive Science Society.

[11] John McCarthy. Programs with common sense. In Marvin Minsky, editor, *Semantic Information Processing*. MIT Press, Cambridge, MA., 1968.

[12] David J. Mostow. Machine transformation of advice into a heuristic search procedure. In Ryszard S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 367–404. Tioga Publishing Company, Cambridge, MA., 1983.

[13] Christopher Owens. Domain-independent prototype cases for planning. In Janet Kolodner, editor, *Proceedings of a Workshop on Case-Based Reasoning*, pages 302–311, Clearwater, FL., May 1988. Defense Advanced Research Projects Agency, Morgan Kaufmann Publishers, Inc.

[14] Michael D. Rychener. The instructible production system: A retrospective analysis. In Ryszard S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 429–460. Tioga Publishing Company, Cambridge, Mass, 1983.

[15] Roger C. Schank. *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Associates, Hillsdale, NJ., 1986.

[16] Donald A. Waterman. Generalization learning techniques for automating the learning of heuristics. *Artificial Intelligence*, 1:121–170, 1970.