Really arguing with your computer in natural language

by MARGOT FLOWERS and MICHAEL G. DYER

University of California at Los Angeles
Los Angeles, California

ABSTRACT

Recently, the computer science community has been hearing a lot about “fifth generation” computers and the Japanese large-scale project to build intelligent software that can “think,” “reason,” and “understand human languages.” It is in the field of artificial intelligence (AI) where such intelligence machines and programs are being designed and created. How far along is the field of AI? How close are AI programs to being able to “reason” or “understand” as humans do? This paper is intended to give scientists outside the field of AI some insight about the problems, issues, and current status of computational models of human argumentation.
INTRODUCTION

As researchers in one subarea of artificial intelligence (AI), called cognitive modeling, we are particularly interested in building models of human cognitive abilities. This paper is intended to give scientists outside of the field of AI some insight about the problems, issues, and current status of one specific area of AI research. In this paper we examine briefly what issues are involved in building a computer capable of engaging in an argument, and then review some past and recent computer programs that have addressed these issues.

MOTIVATION

Before embarking on such an effort, we might first ask why anyone would want to argue with one's own (or anyone else's) computer. First of all, people are always arguing. People argue, for instance, over the U.S. role in Central America, whether the U.S. should restrict Japanese imports, whether God exists, about the importance of the Equal Rights Amendment, and about numerous other topics concerning sex, religion, law, and politics. If we are ever going to have truly intelligent machines, they should, at minimum, be able to understand what's going on in an argument and participate in an argument by defending or justifying a given viewpoint.

Second, as expert systems\(^2\) become more sophisticated, it becomes essential that such systems be able to explain their decision-making processes.\(^3\) In medical expert systems,\(^4\) for example, this involves printing a trace of the rules the system used for arriving at diagnoses. There are many complex domains, however, where the rules are not so clear, and where argumentation and debate are essential to understanding and formulating plans and strategies. Consider a military general, for example: Who would trust the decisions of a general who cannot defend his views in the give and take of a debate?

Finally, understanding or engaging in arguments requires both fundamental reasoning and language comprehension skills. Argumentation turns out to be an ideal task domain for scientifically exploring these basic human capabilities.

A TYPICAL ARGUMENT

Before we can design and implement computational arguers, we must know what problems typically arise in everyday arguments. Consider the following argument fragment, based on the recent destruction by a Soviet military jet of a Korean airliner that strayed into Soviet air space.

[a] Korean: The USSR should be punished for downing the Korean jet.

[b] Soviet: It was a spy plane.

[e] Korean: The U.S. has no need to use a commercial jetliner. Their satellite system is more than adequate.

[f] Soviet: Well, the USSR has a right to protect its borders.

[g] Korean: That may be true, but the USSR has no right to use force against nonmilitary aircraft. Other countries do not shoot down Soviet commercial jets when they stray across national borders.

There is nothing exceptional about this argument fragment. After the Korean jetliner incident, numerous debates arose, many of them similar to this one. We can begin to appreciate the inferential capabilities involved in processing the argument fragment above simply by replacing the 'obvious' responses above with alternative responses.

[a] Korean: The USSR should be punished for downing the Korean jet.

[b] Soviet: It was a spy plane.

[c] Korean: With 269 people on board?

[d] Soviet: They were being used as a cover to protect the U.S. spies hidden on board.

[f] Soviet: Well, the USSR has a right to protect its borders.

[g] Korean: That may be true, but the USSR has no right to use force against nonmilitary aircraft. Other countries do not shoot down Soviet commercial jets when they stray across national borders.

Why is [b] reasonable and not [b1–3]? To avoid generating [b1] for instance, the Soviet must realize that [a] is not a compliment, but an attack. In addition, both the Soviet and Korean must understand the world of international espionage, national borders, and justifications for military force in order to both generate and appreciate the significance of the word “spy” in [b].

[b] Soviet: It was a spy plane.

[c] Korean: With 269 people on board?

[c] is phrased on the surface as a question. However, both argument participants know that [c] does not expect a “yes” or “no” answer. In fact, the Korean is actually saying: “I disagree that it was a spy plane, since 269 spies on board a single plane would be ridiculous. It was therefore a passenger plane; not a spy plane.”
Alternative responses to [f]:

[g1] Korean: ...Other countries do not shoot down Soviet sea gulls when they stray across national borders.
[g2] Korean: ...Other countries do not shoot down American planes when they stray across national borders.

[g1] is perfectly correct, but a totally unreasonable statement. Why? [g2] is also a perfectly true statement, but totally irrelevant to the argument at hand. But how do we know this? Clearly, the Korean is reasoning by analogy in [g] to rebut his opponent’s justification. But how is the correct analogy found?

For a computer to hold up its end in an argument, it must first understand what’s been said. Most arguments are in a natural language; in this case, English. It is doubtful that anyone will find arguments stated in a formal language, such as a mathematical notation or programming language. One can argue about whether a mathematical proof or computer program is correct, but the argument itself most assuredly will be in a natural language.

In addition to understanding the meaning of each natural language statement, the computer must also be able to comprehend the intention of the argument as it unfolds, and how do we keep track of the argument as it unfolds, and how do we represent and apply the argument-so-far as a context for statements yet to follow?

1. How do we represent the meaning of each participant’s statements?
2. How do we encode the world knowledge to which our arguer’s statements are implicitly referring?
3. How do we access these meanings and apply world knowledge in order to go from sentences in the language to these representations of meaning?
4. How do we organize this knowledge so that only the relevant knowledge is applied and the “right” inferences are made at the right time?
5. How do we keep track of the argument as it unfolds, and how do we represent and apply the argument-so-far as a context for statements yet to follow?
6. How do we represent the beliefs of the argument participants?
7. What are the strategies used by argument participants, and how do we represent them?
8. What does the resulting memory of an argument look like?
9. How do factors, such as world knowledge, reasoning ability, memory search, knowledge of language, and argument strategies interact?

There is not enough space here to go into all of these issues in detail, but we will address several of them briefly.

Understanding Language

In the past decade a good deal of progress has been made in text comprehension. Much of this work in natural language understanding has concentrated on the memory structures, inference mechanisms, knowledge representations, and parsing schemes for handling narrative texts. For example, the SAM program used the notion of scripts to read stories concerning stereotypic action sequences, such as going to a restaurant. PAM read stories that dealt with goal and plan relationships between narrative characters. OPUS answered questions about stories involving the use of physical objects. These earlier language-understanding programs were limited to the extent that each dealt only with the application of a single knowledge construct. The next research step involved exploring the adequacy and limitations of these knowledge constructs in the context of more complicated narratives. An important part of this research also involved exploring the way in which multiple sources of knowledge interact during narrative comprehension.

One result of this research effort was a story-understanding program named BORIS. Below is a story fragment read by BORIS:

**Divorce—1**

Richard hadn’t heard from his college roommate Paul for years. When a letter finally arrived from San Francisco, Richard was anxious to find out how Paul was.

Unfortunately, the news was not good. Paul’s wife Sarah wanted a divorce. She also wanted the car, the house, the children, and alimony. Paul...didn’t want to see Sarah walk
off with everything he had. His salary from the state school system was very small. Not knowing who to turn to, he was hoping for a favor from the only lawyer he knew.

Richard eagerly picked up the phone and dialed. After a brief conversation, Paul agreed to have lunch with him the next day. He sounded extremely relieved and grateful.

The next day, as Richard was driving to the restaurant he barely avoided hitting an old man on the street. He felt extremely upset by the incident, and had three drinks at the restaurant. When Paul arrived Richard was fairly drunk. After the food came, Richard spilled a cup of coffee on Paul. Paul was a teacher.

When Paul walked into the bedroom and found Sarah with another man he nearly had a heart attack. Then he realized what a blessing it was. With Richard there as a witness, Sarah’s divorce case was shot.

To understand this story, BORIS must have access to multiple sources of knowledge, including those listed in Figure 1. BORIS coordinates and applies these different knowledge constructs to build an episodic memory of causally connected actions, motivations, settings, interpersonal relationships, and so on. The program then demonstrates its understanding through answering questions. Below are some of the verbatim questions presented to BORIS and the verbatim answers generated:

**What happened to Richard at home?**
Richard got a letter from Paul.

**Why was Paul upset about the divorce?**
Paul and Sarah were fighting over the family possessions.

**What did Paul do for a living?**
Paul was a teacher.

**Why did Paul write to Richard?**
Paul wanted Richard to be his lawyer.

**What happened to Richard on the way to the restaurant?**
Richard almost ran over an old man.

**Why did Richard spill the coffee?**
Richard was drunk.

**How did Paul feel?**
Paul was mad at Richard.

**What happened to Paul at home?**
Paul caught Sarah committing adultery.

In BORIS, all processes of language analysis, knowledge application, and knowledge interactions are implemented as demons. Demons fall within the class of production systems. Demons implement a form of delayed processing and wait until their test conditions are satisfied, at which point they fire and execute their actions. Each live (active) demon is in charge of its own life cycle, deciding how long to stay alive and when to die.

BORIS reads each narrative sentence in a left-to-right order. Entries in the lexicon may be words, phrases, roots, or suffixes. Associated with each lexical item are conceptualizations and attached demons. When a lexical item is recognized, the associated conceptualization is placed into a working memory and its attached demons are spawned.

When demons “fire,” they bind together conceptual structures in working memory and instantiate long-term conceptual structures in episodic memory. These conceptual structures are then accessed by other demons. Thus, both episodic and working memory serve as contexts for parsing. Consider the phrase “picked up the phone and dialed” in “Divorce—1.” In the lexicon, “pick up” is defined in terms of the conceptual dependency conceptualization of GRASP. Associated with this conceptualization are demons that fill in cases associated with GRASP, as in Figure 2. Associated with each unambiguous lexical entry is a single conceptualization. Each unfilled role is followed by an asterisk (*), which acts as a placeholder for a binding. Demons whose task it is to fill these roles appear after the arrow (→). Each arrow indicates where to bind the return values of the demons. Demons that take parameters are enclosed within parentheses, followed by the arguments passed to them.

When BORIS reads “picked up,” the GRASP conceptualization is placed in working memory and the associated demons are spawned. Immediately one of the demons fires and binds George as the ACTOR of the GRASP. When “phone” is encountered, another instantiation of the same EXPECT

**Associated demons:**

**EXPECT [Pattern, Direction]**
Search Working Memory for Pattern in the Direction specified When found, bind to role

**APPLY-KS [ACT]**
If a primitive CD ACT is encountered
Then examine the OBJECT of the ACT
and If the OBJECT has an associated script or MOP
Then apply that script of MOP to the ACT
If MOP found which is uninstantiated,
Then create an instance in episodic memory

**Figure 2—Demons associated with “GRASP”**
demon fires and binds PHONE as the OBJECT of the GRASP. At this point, APPLY-KS fires. It is the task of this demon to reinterpret the GRASP in terms of a larger knowledge structure. APPLY-KS contains several heuristics. One heuristic is to search BORIS's object-primitive knowledge of whatever is bound in the OBJECT slot. Through the physical object of PHONE, BORIS accesses the knowledge structure M-PHONE, which holds information about how to answer and make calls. APPLY-KS then applies M-PHONE to the GRASP conceptualization. Since GRASP(phone) is an act in M-PHONE, the match succeeds. Since this is the first instance of a phone call with Richard as the caller, BORIS creates an instantiation of this event in episodic memory with Richard as CALLER. When “dialed” is read, the demon associated with dialing will immediately find M-PHONE instantiated in working memory with a pointer to a corresponding event in episodic memory. As a result, the dialing action will simply update this instance of M-PHONE. Meanwhile, demons associated with M-PHONE have been spawned. These demons look for the recipient of the call, the message being conveyed, and whatever goal Richard plans to achieve by making this call.

Thus, the process of comprehension may be abstractly characterized as a cycle of processes that build new knowledge structures, where both demons and knowledge structures arise from lexical input.

The research described here is based on the premise that natural language comprehension is a process of intelligent inference, memory access, and knowledge application. Comprehension requires a great deal of prior world knowledge. The key to understanding lies in computational insight about human knowledge and memory constructs: their representation, application, instantiation, interaction, control, coordination, indexing, access, search, and retrieval. For each class of knowledge there are associated different processes of memory search, inference, and language analysis. Knowledge constructs also interact with one another in different ways. For example, knowledge of emotions will interact with goal and plan situations, for example, the failure of a plan or goal may cause frustration or anger.

The knowledge constructs used by current story-understanding and question-answering systems serve as a necessary but insufficient foundation for dealing with arguments. Although argument participation requires knowledge of goals, plans, scripts, themes, and so on, additional knowledge and processing constructs are required.

Representing Beliefs About the World

In an argument, the arguer defends his own beliefs while attacking those of his opponents. Understanding the conceptual content of an argument, therefore, requires the computer to possess a representation of opposing ideologies.

Two researchers who have worked on computational models of political beliefs are Abelson and Carbonell. Abelson developed a model of conservative political ideology called the “Cold Warrior,” which was intended to capture Sen. Barry Goldwater's belief system. The most important notion in Abelson's work was the master script, which modeled relationships among various themes in a conservative Cold War ideology, as depicted in Figure 3. Sequences of interactions and conflicts among themes were organized by answer frames. For example, one ideologically conservative answer frame contained the sequence given in Figure 4.

The master script notion was very useful for capturing the relationships among several of Goldwater's beliefs. Using this master script, Abelson's “Cold Warrior” could express its conservative point of view. Carbonell has pointed out that the Cold Warrior lacked a knowledge of mundane reality. In addition, while capable of expressing a point of view, the Cold Warrior model was incapable of defending it. In contrast, the real senator can support his own beliefs and can refute the attacks and criticisms of his opponents, using counterexamples and various forms of reasoning and argumentation.

In his program, POLITICS, Carbonell modeled political ideologies in terms of goal trees. Goal trees both organize a set of goals according to subgoal relationships and ranked them in terms of their relative importance. For example, the goal tree fragments in Figure 5 represent a portion of a U.S. conservative ideology in Carbonell's model. In addition to their own set of goals, the liberal and conservative each maintain goal trees representing perceived Soviet ideologies. For example, the conservative believes that the Soviet's most important goal is Western domination, while the liberal does not.

Carbonell used goal trees combined with counterplanning strategies in POLITICS to model subjective understanding of political events. The POLITICS program could predict events and answer questions by accessing the goals and counterplanners of the characters involved. For example, if asked the question: “Should character X do act Y?” POLITICS then checked the consequences of the act to see if it might achieve one of the character's goals. If asked a question of the form: “What should character1 do if character2 does act X?” the program

When: Communist Schemes, then: Communist Victory, unless: Use of Free World Power, in which case: Free World Victory.

Figure 4—Conservative answer frame
checked whether the act blocked character1's goals, and if so, it accessed an appropriate counterplan to formulate an answer.

Input Story: The U.S. Congress is expected to approve the Panama Canal treaty.

(Interpret U.S. — Conservative)
Q: Should the U.S. approve the treaty?
A: No, the treaty is bad for the United States.
Q: Why is the treaty bad for the U.S.?
A: The United States would lose the canal to Panama and the United States will be weaker.
Q: Why should the U.S. worry about Russia taking over the canal?
A: Because Russia wants to control the world.

(Interpret U.S. — Liberal)
Q: Should the U.S. approve the treaty?
A: Yes. The Panama Canal Treaty is good for the United States.
Q: What if Russia tries to take over the Canal?
A: Russia does not want conflict with the U.S. . . .

Applying Argument Strategies

Recently, research has been done in building process models of argumentation. One computer program, ABDUL/ILANA, and can model either side of an argument between an Arab (ABDUL) and Israeli (ILANA) over who was responsible for the 1967 Arab-Israeli war. This argument dialogue appears below:

[a] Arab: Who started the 1967 War?
[b] Israeli: The Arabs did, by blockading the Straits of Tiran.
[c] Arab: But Israel attacked first.
[d] Israeli: According to international law, blockades are acts of war.
[e] Arab: Were we supposed to let you import American arms through the Straits?
[f] Israeli: Israel was not importing arms through the Straits. The reason for the blockade was to keep Israel from importing oil from Iran.

[g] Arab: But Israel was importing arms, and that's because Israel is trying to take over the Middle East.

[h] Israeli: If Israel were trying to take over the Middle East, then why didn't Israel take Cairo in 1973? Major issues addressed in ABDUL/ILANA were (1) How does X even recognize that Y's opponent Y has attacked a belief of X, or has supported one of Y's beliefs? (2) Given an attack by Y, how does X choose a defense? and (3) How is an argument represented in memory? We will briefly discuss these issues using the argument dialogue above.

We can see that each participant here is accusing the other participant's notion of starting a war. To recognize that [b] is an accusation, ABDUL must know that wars are "bad," that parties responsible for causing bad events to occur are also bad, that accusations involve attributing responsibility for a morally reprehensible event, that any accusation should receive a rebuttal if possible. To counter the accusation in [b], ABDUL searches its memory of historical events and notices that Israel fired first. This fact forms a useful rebuttal, but why? There are other events in historical memory, such as resolutions before the United Nations, results of Israeli elections, and so on, yet none of these constitute a useful rebuttal. ABDUL must know general facts, such as: (1) a war contains a sequence of events and (2) the initiator of the first event in certain sequences of events can be viewed as responsible for the entire sequence.

In [c], the Arab asked the Israeli a question that ostensibly requires an answer of "yes" or "no." However, ILANA must realize that neither answer is appropriate; that the question is actually rhetorical, and that it contains an indirect accusation. In [d], the Israeli directly denies the accusation made in [c] and counters with another accusation. Notice that in [f] and [g] the argument revolves around whether oil or arms were being imported into Israel. As readers we realize that this distinction is highly relevant, but how? To recognize the relevance of this distinction, ABDUL/ILANA must possess knowledge about the world of foreign trade and of armaments. Furthermore, ABDUL/ILANA also must possess more abstract knowledge of how goals and plans interact with argument strategies. This knowledge includes rules, such as:

IF X believes that Y is executing action A to enable X to violate a goal G of Y
THEN Y may block A and use the fact of X's intention as a justification for Y's action

In [g], the Arab accuses the Israeli of "trying to take over the Middle East." The Israeli's rebuttal in [h] is very effective, but how did ILANA come up with this, especially since the event of taking Cairo never happened? Clearly, there are an infinite number of nonevents, so storing them all in memory is simply impossible. In this case, ABDUL/ILANA contains a theory of expectation failures, which organize memory around predicted and unpredicted events. Since the take-over of Cairo was predicted but unfulfilled, it is stored explicitly in memory as an expectation failure. Such failures are then used in arguments as one source of counterexamples.
As the argument progresses, ABDUL/ILANA builds up a conceptual, language-independent representation of the argument, called an argument graph (or a-graph). 18,23 The a-graph is composed of beliefs connected by relationships of attack (a) and support (s). Figure 6 is a simplified fragment of the a-graph for part of the argument over the Six-Day War. Whenever ABDUL/ILANA received an opponent's input, it searched the argument graph to determine what belief was being supported or attacked. For example, a belief could be attacked by a direct contradiction, such as:

Israeli: The Arabs are responsible for the war.
Arab: No. The Israelis are responsible for the war.

However, direct attacks are not very effective. Furthermore, use of direct contradiction by both opponents leads to the "Did!" "Didn't!" "Did!" "Didn't!" argument "loop" that children's arguments often exhibit. A more effective approach is to attack one's opponent by finding support for a claim that contradicts a claim of the opponent. By using this strategy, ABDUL determines that the following response is more suitable:

Israeli: The Arabs are responsible for the war.
Abdul: But Israel fired first.

Reasoning During Argumentation

People support their beliefs with chains of reasoning. The HARRY program 24 explores the role of reasoning during arguing. A fundamental notion in HARRY is that human reasoning frequently makes use of previous examples, prior chains of reasoning formed from previous arguments, and adaptations of situations related to the current problem. Human reasoning rarely relies on the application of general-purpose rules. Thus, reasoning in HARRY is viewed as a memory-based process. In general, turning what was a rule-based task into a memory-based task makes the process of comprehension easier.

HARRY has been used to model various forms of reasoning in economic and political domains. Consider the following transcript, taken from an interview with the economist, Seymour Melman, on the “MacNeil/Lehrer Report”:

“Economist: The increased funding of DOD [the Department of Defense] has lead to today's higher rate of inflation in the U.S.”

Machinery producers customers of the government
Cost maximizing became a part of machinery producers
Costs of production and administration increased
Prices of new machinery rose
New machinery purchase no longer desirable
Prices of new machinery increased
Productivity growth rate falls
...
Inflation increases

“Interviewer: Why do you believe that?”
“Economist: Well, DOD is publicly funded. As a result, they tend to cost maximize. That is, they do not carefully monitor their costs. Starting in the fifties, DOD became a major customer of machinery-producing industries. Thus, the heavy machinery industry began to cost maximize as well, causing the prices for new heavy machinery to increase. As the purchase of heavy machinery decreased, the mechanization per worker decreased, causing the productivity growth rate to decrease as well. As a result, inflation has increased.”

The reasoning chain produced by Melman is rather complicated and includes the components depicted in Figure 7. How might Melman have produced this chain of reasoning? Producing this particular chain through the application of very general rules is possible but would be very costly. Clearly, Melman had already produced identical, or at least related, chains of reasoning before appearing on “MacNeil/Lehrer.” Instead of applying general reasoning rules, experts such as Melman more likely retain previous reasoning chains in memory. These chains are generalized, chunked, indexed in various ways in memory, and then subsequently retrieved, adapted, and applied.

The first time HARRY produced the above chain of reasoning, it was done laboriously through the application of general rules. The resulting chain was then abstracted to form reasoning scripts (e.g., R-GOV-SPEND→INFLATION). These reasoning scripts are then applied to similar reasoning problems. Thus, HARRY’s reasoning becomes more memory-based and less rule-based as HARRY goes through successive reasoning experiences.

CONCLUSIONS

Traditionally, reasoning has been characterized as a process of logic. This is not surprising since some of the first people to address reasoning were logicians. Also, a common tactic in arguments and editorials is to accuse one’s opponent of being
“illogical.” However, protocols indicate that people do not reason by syllogism, nor do they employ carefully constructed proof-style chains of deductive reasoning.

Whether an arguing system is implemented in a logic-programming language, such as PROLOG,25 or a functional programming language, such as LISP,26 the choice made is actually neutral toward the problems raised in modeling human argumentation. The most interesting issues that arise during everyday arguments are not logical as such, but rather involve problems in representing, organizing, and applying human knowledge constructs. The limitations of our programs reflect our limited theoretical understanding in these areas. Meanwhile, it will be some time before our home computers start arguing with us or requiring us to possess Socratic debating skills just to talk them into executing our home income tax programs.

ACKNOWLEDGMENTS

At UCLA, the work of both authors has been supported in part by the National Science Foundation. Part of the work described here was performed while the authors were at Yale University, supported in part by the Office of Naval Research, in conjunction with Larry Birnbaum, Pete Johnson, Rod McGuire, and Tom Wolf.

REFERENCES
