In the News

ALSO FEATURED

THE BERKELEY PARSER: HIGH-QUALITY GRAMMAR, AUTOMATICALLY

Features Editor: **Dennis Taylor**
dtaylor@computer.org

Multiagent System Helps Train for the Unthinkable

Mark Ingebretsen

The UK terrorist attacks this June demonstrated once again the extremists' preference for hitting multiple targets. This strategy aims to confuse authorities and increase casualties and damages. If a large city becomes the target of multiple attacks, responders will need to scramble to allocate fire trucks, ambulances, and other resources to any number of locations. Meanwhile, they'll need to balance factors such as the intensity of the resulting fires or bomb-related damage at each location, as well as traffic and wind patterns throughout the city. And they'll need to consider the possibility that more attacks might be imminent.

If ever a problem was in need of an AI-based simulation, this is it. And that's just what DEFACTO (Demonstrating Effective Flexible Agent Coordination of Teams through Omnipresence) aims to provide.

Why AI-based simulation?

So far, training for such a nightmare scenario has been difficult, says Ronald Roemer, a captain and 32-year veteran of the Los Angeles Fire Department. Conducting a large-scale live exercise is "very time consuming and expensive," he says. "You'd have to have a lot of players to play the different parts of the individuals who are responders so that you'd have enough inputs to challenge the incident commander [the official in charge of directing the response]."

Those difficulties can be compounded by the fact that responders have scant experience dealing with large-scale multiple emergencies, let alone terrorist attacks, simply because such events (thankfully) remain rare.

As a workaround, some years ago, Roemer and his LAFD colleagues began building a simulation facility for disaster training. Meanwhile, they searched for training software that could simulate a citywide terrorist scenario. What they needed, Roemer explains, was a system that could communicate reams of information about each emergency location to challenge the incident commander and evaluate how well he or she could support responders on the scene.

Ideally, any AI application for simulating such a volatile situation would allow flexible delegation of authority. Emergency crews on the ground need some autonomy to execute their plans quickly. Likewise, incident commander trainees will get bogged down if they try to micromanage the response at each location.

Conversely, trainees need feedback to see whether emergency crews' efforts are succeeding or additional resources are necessary. So, any AI application would need to let trainees step in and help at a scene if needed and to establish unimpeachable communication pathways to ensure they could intercede. It would also need to provide macro and micro views of the events happening.

DEFACTO

What Roemer and his colleagues found was DEFACTO, an application that was being developed by Nathan Schurr, then a doctoral student at the University of Southern California (http://teamcore.usc.edu/schurr/research.shtml). DEFACTO can't yet completely recreate a multi-incident terrorist attack scenario. Indeed, months of work are still necessary before that occurs. But even in its unfinished state, the program has a realistic visual interface and the ability to model aspects of the devastation from such an attack.

DEFACTO's Omni-Viewer interface gives trainees two views of a disaster scenario. The overview consists of a 2D map of LA that shows the locations of fires and the assets responding to them. A second, 3D view zooms in for an interactive representation of streets and actual buildings at a disaster scene, duplicating with computer graphics what responders located there see.

Roemer has been providing Schurr and the other designers background on the LAFD's basic operations. He also has provided feedback on one of the program's more developed scenarios, a high-rise fire. The department's response to a high-rise fire, as with any other major emergency, would follow a chain-of-command structure

1541-1672/07/$25.00 © 2007 IEEE
Published by the IEEE Computer Society

originally developed by the military, he says. In this structure, "task level" personnel at one or more disaster scenes are charged with quelling the fires. Meanwhile, the incident commander, at an operations center away from the scene, directs the LAPD's response throughout the city. The incident commander in turn is assisted by "individuals charged with overseeing sections such as operations and logistics," Roemer says.

In lieu of living, breathing emergency personnel, DEFACTO employs a multiagent system that helps direct the movement of firefighting resources and how each resource tackles its assignments. This system attempts to unite software agents, robots, and humans into cohesive teams—with the required delegation of authority and seamless communication essential to addressing multiple emergencies.

As Schurr explains: "If multiple fires broke out in LA, there would be the overall objective to extinguish them all. Then distinct plans would be generated for each of the fires. And consequently there would be certain roles that would be associated with each of these plans. The challenge for the team would be to take their limited resources and throw them at an overwhelming amount of tasks to fight fires."

In a DEFACTO simulation, those charged with executing the individual firefighting plans might be humans or nonhuman agents simulating specific emergency assets. An example of the latter might be a robotic bomb retrieval unit. Like any human given an assignment, the robot would manage the specifics of its assigned tasks by relying on its own intelligence. Supervision of the robot by the trainee would be broad based, not task specific. That is, if the robot was failing at its task, trainees might pull it from the scene or send in an additional robot to help. But they wouldn't control how it searched a building or disposed of a bomb.

A team of personal assistants

To help create this delegation of authority, DEFACTO incorporates a layer of *proxies* between the incident commander and the team members, says Schurr. Proxies essentially are intelligent personal assistants; one is assigned to each agent.

"Our proxies have social intelligence," says Milind Tambe (http://teamcore.usc.edu/tambe), computer science professor at the University of Southern California and Schurr's doctoral advisor. As he explains it, the proxies that form part of DEFACTO are each programmed with the same commonsense rules of teamwork, including how and when to communicate and to assist teammates requiring help.

Because the proxies' functionality is deliberately limited to team building, programmers needn't spend time imbuing them with knowledge about teamwork, says Tambe. "We've raised the level of abstraction in the program from the programming of individual agents to a team-level abstraction," he says. So, emergency responders practicing with DEFACTO can focus on the big picture.

By the same token, the teams can self-organize around the task of firefighting in the most efficient way possible. Obviously, humans need to be able to make final decisions if necessary. But human input isn't required.

Tambe says that DEFACTO allows for "a completely autonomous mode where the fire engines proxies, without any human intervention, can allocate themselves to the best of their ability to the fires. But we also allow for a different mode where it's all right when there's a difficulty to involve a human. So essentially, when there are tasks that they're unable to perform because there just aren't enough of them, they can say, 'We need help on these tasks.' The human can intervene, and the proxies keep adjusting to the tasks depending on the level of intervention."

Machinetta

The proxies employ Machinetta, a team-oriented programming architecture (free at http://teamcore.usc.edu/doc/Machinetta) that Tambe codeveloped with Schurr and Paul Scerri, currently a systems scientist at Carnegie Mellon University (www.cs.cmu.edu/~pscerri). Machinetta's script is written in Java, to facilitate cross-platform deployment.

As Schurr and his colleagues explain in a paper on DEFACTO (http://teamcore.usc.edu/papers/2005/SS105SchurrN.pdf), Machinetta comprises five modules, each containing the code for a specific proxy behavior. One module contains algorithms for adjusting the team members' level of autonomy. The others control such things as coordination reasoning along with communication between proxies and between individual proxies and team members. Machinetta lets users insert additional algorithms governing interactions between proxies as plug-and-play units.

IEEE Intelligent Systems

How to Reach Us

Writers

For detailed information on submitting articles, write for our Editorial Guidelines (isystems@computer.org) or access www.computer.org/intelligent/author.htm.

Letters to the Editor

Send letters to

Dennis Taylor, Lead Editor
IEEE Intelligent Systems
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
dtaylor@computer.org

Please provide an email address or daytime phone number with your letter.

On the Web

Access www.computer.org/intelligent for information about *IEEE Intelligent Systems.*

Subscription Change of Address

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify *IEEE Intelligent Systems.*

Membership Change of Address

Send change-of-address requests for the membership directory to directory.updates@computer.org.

Missing or Damaged Copies

If you are missing an issue or you received a damaged copy, contact membership@computer.org.

Reprints of Articles

For price information or to order reprints, email isystems@computer.org or fax +1 714 821 4010.

Reprint Permission

To obtain permission to reprint an article, contact William Hagen, IEEE Copyrights and Trademarks Manager, at copyrights@ieee.org.

IEEE Intelligent Systems

IEEE Computer Society Publications Office

10662 Los Vaqueros Circle, PO Box 3014
Los Alamitos, CA 90720-1314

A DEFACTO simulation.

Future challenges

At present, Machinetta doesn't enable proxies to self-learn, although that remains a goal of Tambe and Schurr. Incorporating self-learning that lets DEFACTO provide better guidance to trainees is especially challenging, because large-scale disasters are unique. For example, says Schurr, "little is known about what the utility of putting out a particular fire for the whole team," However, "By running numerous training simulations, the system could learn that putting out a certain fire would provide the team with a quantifiable benefit."

A longer-term goal would be to use DEFACTO also as an assistant in an actual emergency. Innumerable inputs aimed at providing incident commanders with vital information could be added, says the LAFD's Roemer. Examples might include improved traffic control as units speed toward a disaster site, a built-in database on the composition and layout of affected buildings, and video cameras to supplement the computer-generated view. The latter would be especially useful because it would give incident commanders an actual picture of events as they transpire.

Meanwhile, in the months ahead, Tambe and his colleagues, working with their LAFD advisors, have a more modest, though certainly invaluable, goal. They want to perfect DEFACTO so that it can provide incident-commanders-in-training with an authentic simulation of a large-scale terrorist attack, as one way to prepare their minds for unthinkable events.

The Berkeley Parser: High-Quality Grammar, Automatically

Maya Dollarhide

When Daniel Klein was an undergraduate, he wasn't sure how he would meld his two favorite subjects—linguistics and computer science and programming. Today it's how he makes his living.

Klein is an assistant professor in the Electrical Engineering and Computer Sciences Department at the University of California, Berkeley. "In general," he says, "I work on algorithms and systems for analyzing and processing natural languages. This includes end applications like machine translation, information extraction, and speech recognition, as well as core tools like syntactic parsers, which are needed by many applications." He holds degrees in linguistics and computer science. "Linguistics," says Klein, "helps us understand the deep and rich structures of human languages; machine learning helps us build robust, accurate models of those structures. And then it's an engineering task to make the implementations efficient."

These combined interests eventually led to the Berkeley Parser, which automatically

learns a language's grammar and then can determine the most likely structure of a sequence of words in that language. The parser was developed by Klein and his team, led by graduate student Slav Petrov, at the Berkeley Natural Language Processing Group (http://nlp.cs.berkeley.edu/Main.html). For his research on the unsupervised learning of grammars, Klein won the Association of Computing Machinery's 2006 Grace Hopper Award.

The challenge of automatic learning

Given a sequence of words, a parser identifies parts of speech such as subject, verb, and object, and then determines the relations between them. Once a parser has learned a language's grammar, it systematically predicts parse trees (structure diagrams) and eventually arrives at a final parse.

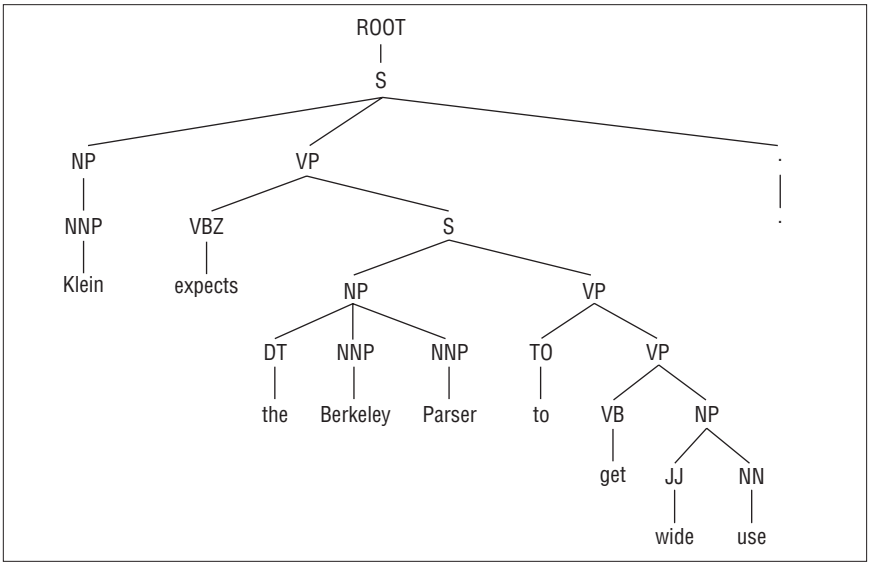
The Berkeley Parser constructs a grammar by generalizing human-provided examples from tree banks. (A tree bank is a collection of sentences whose syntactic structures are represented as tree diagrams.)

According to Klein, the parser efficiently and compactly represents all the possible parses of a sentence. It then effectively and compactly considers the best parse according to its model (grammar) and its error function. "The parser also 'prunes' while it goes, so it gets rid of any unlikely parses in a certain hierarchical fashion while it is choosing the best parse," says Klein. "To the extent that the models—grammars here—are good, the system makes the optimal choices," he says.

Klein says that the challenges unique to his parser were trying to make a system that was (1) simpler, (2) more automated, and (3) more accurate than past methods.

"We knew (1) was possible, and even that (1) and (2) together should be possible. But we had no idea whether all three were possible. It turned out that, as is often true, the right elegant system works better than baroque solutions," he says.

Klein adds the team knew they were onto something when they discovered that they could use a classical incremental learning method with the parser. By employing hierarchical coarse-to-fine learning, the parser would be able to go through the options from the treebank and efficiently select the final best choices. From there it could then come up with a final prediction, with little overhead.



An example parsed sentence from the Berkeley Parser, which automatically learns a language's grammar and determines the most likely structure of a sequence of words.

Elegance in parsing

The publicly available system that's most similar to the Berkeley Parser is Eugene Charniak and Mark Johnson's Brown Parser, which has been constantly improving and evolving for years. Klein remarks that the Brown Parser can "take any system, including his" because it's also fast and accurate—"the best of the lexicalized systems." (Lexicalized parsers give each word its own syntax; the Berkeley Parser takes a broader approach by categorizing words into syntactic classes.)

Charniak agrees with Klein. "My system may be more accurate, but Klein's is much more creative and elegant," he says. "And Klein's is not too far behind ours, in the sense that maybe our parser will have seven out of 100 errors and Klein's might have eight and a half. … Those numbers are hypothetical, but you get the idea."

According to Charniak, one thing that makes the Berkeley Parser great is that it actually figures out how to analyze sentences that it needs to parse, whereas other parsers depend on already having been programmed with information that proves to be statistically correct for various languages. "The Berkeley Parser suggests improvements in translations that have not been entered into the system, and that is one reason why it's an amazingly elegant parser," says Charniak. "I call it prettier or more elegant than others because the parser is actually learning more by itself—whether or not it's always accurate is less important. It deserves every award that it gets."

What's in the future

Klein expects the Berkeley Parser to get wide use. He says that people can use it to do whatever they're doing with other parsers, only faster and more accurately. "This includes information extraction, language modeling, machine translation, whatever," he says. Currently the parser can handle German, Chinese, and English. There are tree banks for dozens more languages, says Klein, but they haven't tested the system on them all.

According to Klein, the online demo (at http://nlp.cs.berkeley.edu/Main.html#research_overview) "has some classroom applicability—for example, in natural language processing and linguistics classes."

The Berkeley Natural Language Processing Group is also using the Berkeley Parser to improve machine translation. "We're working with the machine translation group at ISI [the University of Southern California Information Sciences Institute] on integrating our parser into their top-performing system, and we're building our own Berkeley MT system around the parser and the algorithms behind it," says Klein.

Charniak says he believes the field will soon see parsing used in speech recognition programming. "It's in the experimental stages right now, but it will happen," he says. "But that's a whole other story." ■