Deploying Information Agents on the Web

Craig A. Knoblock
University of Southern California
Information Sciences Institute and Computer Science Department
4676 Admiralty Way
Marina del Rey, CA 90292, USA
knoblock@isi.edu

Abstract

The information resources on the Web are vast, but much of the Web is based on a browsing paradigm that requires someone to actively seek information. Instead, one would like to have information agents that continuously attend to one's personal information needs. Such agents need to be able to extract the relevant information from web sources, integrate data across sites, and execute efficiently in a networked environment. In this paper I describe the technologies we have developed to rapidly construct and deploy information agents on the Web. This includes wrapper learning to convert online sources into agent-friendly resources, query planning and record linkage to integrate data across different sites, and streaming dataflow execution to efficiently execute agent plans. I also describe how we applied this work within the Electric Elves project to deploy a set of agents for continuous monitoring of travel itineraries.

1 Introduction

There is a tremendous amount of information available on the Web, but the access to this information is largely dependent on the information providers. The individual web sites decide what information to provide to the visitors to their site and how a visitor will access that information. There is little in the way of capabilities to combine information across sites and only the most primitive capabilities to monitor information on individual sites. Of course, sites may provide integrated access to specific data sources or sophisticated monitoring capabilities on their data, but users of the Web are dependent on a site to make these capabilities available.

In contrast, imagine a world where access to information is driven by the consumers of the data. Where it is not only possible, but simple to task your own personal information agents to gather specific information, monitor for changes in the gathered data, and to notify you of important information or changes. The challenges are that the required information is often embedded in HTML pages, the data is organized in different ways on different sites, there are a huge number of ways the information could be combined, and it can be slow and cumbersome to combine the information.

We have developed a set of core technologies to simplify the task of building intelligent agents for gathering and monitoring information on the Web [Knoblock *et al.*, 2003; 2001; Barish and Knoblock, 2002a]. The technologies include the ability to gather information from the Web, to link that information with related information, to build plans to integrate the various sources of data, and to efficiently execute these plans in the Web environment. We have applied these technologies to build agents for a variety of applications [Barish *et al.*, 2000; Ambite *et al.*, 2002], including an application for monitoring travel plans from online sources.

Researchers have developed a variety of agents that have been deployed on the Web over the years. Some notable systems include the Internet Softbot [Etzioni and Weld, 1994], an agent that interacts with a range of Internet resources, BargainFinder [Krulwich, 1996], a comparison shopping agent for CDs, ShopBot [Perkowitz *et al.*, 1997], a system for automatically locating and incorporating new stores into a comparison shopping agent, and Warren [Decker *et al.*, 1997], a system for gathering data on a financial portfolio. There has also been a significant amount of research on the underlying technologies required for developing agents on the Internet [Levy and Weld, 2000]. As noted by Etzioni [1996], the Web provides a tremendous opportunity for building intelligent software agents. Yet, surprisingly few have seized this opportunity. This is almost certainly because there are many technical issues that must be addressed to build such agents. Our work is designed to address these issues and simplify the task of building agents for the Web.

In this paper I first describe an example set of agents for monitoring travel plans. Then I will briefly describe the technologies that we have developed to gather data from web sources, link data across sources, generate plans to integrate the data, and efficiently execute these plans. Finally, I present directions for future research and conclusions.

2 Information Agents for Monitoring Travel

As part of the Electric Elves project [Chalupsky *et al.*, 2001; Ambite *et al.*, 2002] we have applied our agent technologies to build a set of agents for various tasks including tracking visitor schedules, monitoring meeting schedules, and monitoring a user's travel plans. In the case of monitoring travel plans, this task is particularly well-suited for applying agent technology for several reasons: a) this is a fairly compli-

cated task with many possible forms of failure ranging from flight cancellations and schedule changes to hotel rooms being given away when a traveler arrives late at night, b) there are a large number of online resources that can be exploited to anticipate problems and keep a traveler informed, and c) these tasks would be tedious and impractical for a human to perform with the same level of attention that could be provided by a set of software agents.

To deploy a set of agents for monitoring a planned trip, the user first enters the travel itinerary and then specifies which aspects of the trip she would like to have the agents monitor. A set of information agents are then spawned to perform the requested monitoring activities. For the travel planning application, we developed the following set of agents to monitor a trip:

- A airfare-monitoring agent that tracks the current price of a flight itinerary. The agent sends a notification on price increases and/or decreases. A traveler might consider reticketing if the price drops significantly below what they paid.
- A schedule-change agent that keeps track of the published schedule for a given flight itinerary and notifies a traveler of any change to this itinerary. Small schedule changes occur quite frequently, especially if one purchases tickets well in advance of a flight. Travel agents are supposed to notify their customers of changes, but one usually arrives at the airport before discovering that the scheduled departure time has changed.
- A flight-status agent that continually monitors the status of a flight. When a change of status or cancellation is detected, the traveler is immediately notified. This agent also sends a fax to the hotel if the flight arrival is delayed past 5pm in order to ensure that the hotel room is held for the traveler. This agent differs from what is available from commercial sites, such as ual.com, which simply check the status a fixed period of time prior to the flight. In contrast, our agents maintain state and can notify the user of multiple status changes without sending messages that provide no new information.
- An earlier-flight agent that checks for flights that leave before the scheduled flight. It also checks the status of these flights to avoid suggesting delayed or cancelled flights. This agent is particularly handy when one finishes a meeting early or one wants to skip out of a particularly boring meeting.
- A flight-connection agent that monitors a users scheduled flights and if there is a connecting flight it wakes up a few minutes before the projected landing time, checks the status and gate information of the connecting flight and also searches for any earlier flights to the same destination that the user might be able to take instead. This agent is particularly useful when there are only a few minutes to make an earlier connection that is about to depart.
- A restaurant-finding agent that locates the nearest restaurant based on the user's GPS location. On request, it suggests the five closest restaurants providing cuisine type, price, address, phone number, latitude, longitude, and distance from the user's location.

These agents are scheduled to run at regular intervals, where the agents are woken up to perform their task. The agents can cancel their own task once it is complete and can change the interval in which they are run based on the information from other agents. The agents often invoke other agents to help them perform their tasks. For example, the flight-status agent calls another agent that extracts the flight status information directly from a web site and it invokes the hotel notification agent, which in turn sends a message to the fax agent.

Figure 1 shows the messages that various agents generated during actual use of the system. The original set of agents were in use for about a year and then based on feedback and requests from the users we recently developed a new set of agents that provide improved capabilities.

(a) *Airfare-Monitoring Agent*: Airfare dropped message

The airfare for your American Airlines itinerary (IAD - LAX) dropped to $281.

(b) *Schedule-Change Agent*:

The schedule of your United Airlines flight 1287 has changed from 7:00 PM to 7:31 PM.

(c) *Flight-Status Agent*: Flight delayed message

Your United Airlines flight 190 has been delayed. It was originally scheduled to depart at 11:45 AM and is now scheduled to depart at 12:30 PM. The new arrival time is 7:59 PM.

(d) *Flight-Status Agent*: Flight cancelled message

Your Delta Air Lines flight 200 has been cancelled.

(e) *Flight-Status Agent*: Fax to a hotel message

Attention : Registration Desk

I am sending this message on behalf of David Pynadath, who has a reservation at your hotel. David Pynadath is on United Airlines 190, which is now scheduled to arrive at IAD at 7:59 PM. Since the flight will be arriving late, I would like to request that you indicate this in the reservation so that the room is not given away.

(f) *Earlier-Flight Agent*:

The status of your currently scheduled flight is:

190 LAX (11:45 AM) - IAD (7:29 PM) 45 minutes Late

The following United Airlines flight arrives earlier than your flight:

946 LAX (8:31 AM) - IAD (3:35 PM) 11 minutes Late

(g) *Flight-Connection Agent*:

Your connecting United Airlines flight 925 will depart at 9:45 PM (25 minutes late) at gate C6.

(h) *Restaurant-Finding Agent*:

These are the five closest restaurants from your location.

Wingmaster's on I St, American, 1825 I St NW, 202-429-0058, $5-10, Lat: 38.90111, Lon: -77.04158, 0.23 miles

...

Figure 1: Actual messages sent by monitoring agents

3 Gathering Data from Web Sources

A key capability for information agents is the ability to reliably access information. As the Web moves towards XML and Web Services, accessing data could become greatly simplified. However, movement in this direction has been quite

slow and for various reasons many sources will remain available only in HTML, so there is still a critical need to turn HTML sources into agent-enabled sources.

The challenge in building wrappers for online sources is how to achieve broad coverage and high accuracy with minimal user input. The two general approaches to this problem are supervised machine learning techniques [Kushmerick, 1997; Muslea *et al.*, 2001; Hsu and Dung, 1998] and unsupervised grammar induction techniques [Lerman *et al.*, 2001; Crescenzi *et al.*, 2001; Hong and Clark, 2001]. The unsupervised grammar induction techniques have the advantage of no user input, but they are not able to handle the full range of semistructured sources. In contrast, the supervised learning techniques apply to a wider set of sites, but can require a significant amount of labeled data to achieve high accuracy.

We developed an machine learning algorithm called Stalker [Muslea *et al.*, 2001] that requires labeled data, but attempts to minimize the amount of information that must be provided by a user. Given labeled data, the system employs a greedy set covering algorithm to learn extraction rules that define a wrapper for a source. We minimize the amount of labeled data required by decomposing the learning problem into a number of simpler subproblems, which require fewer examples to learn. The decomposition is based on the hierarchical structure of the information in a web source. This approach allows Stalker to learn how to extract data from complicated sites that involve lists of information and even arbitrary nesting of embedded lists.

An issue for any learning system, even Stalker, is that to achieve high accuracy the system must see the right set of examples. Since the expectation in a wrapper is to extract the data with 100% accuracy, finding a representative set of examples is a critical part of the problem. Rather than relying on the user to identify these examples, we developed an active learning technique called Co-Testing [Muslea *et al.*, 2000; Muslea, 2002; Muslea *et al.*, 2003] that selects the most information examples to label. Co-Testing works by learning multiple classifiers using different views of the same problem. In the case of wrapper learning, the system exploits the fact that it can learn equally well a classifier by finding landmarks from the beginning of the page or by finding landmarks from the end of the page. The system can then exploit the fact that both classifiers should agree if they have learned the same concept and any disagreement provides a source of training examples. Both classifiers are applied to the unlabeled examples and the user is asked to label the examples where there is disagreement. This allows the system to quickly identify the unusual cases in the data set to rapidly converge on an accurate set of extraction rules.

Another important challenge building wrappers is to ensure that they continue to work properly over time. This problem has not received much attention. The exception is the work by Kushmerick [2000] who developed an approach that uses the global properties of a page, such as the density of HTML tokens on a page, to determine when the page or even the specific information being extracted has changed. The limitation of this approach is that it is too coarse to detect some sites that have changed [Lerman *et al.*, 2003].

We developed a wrapper maintenance system that can re-

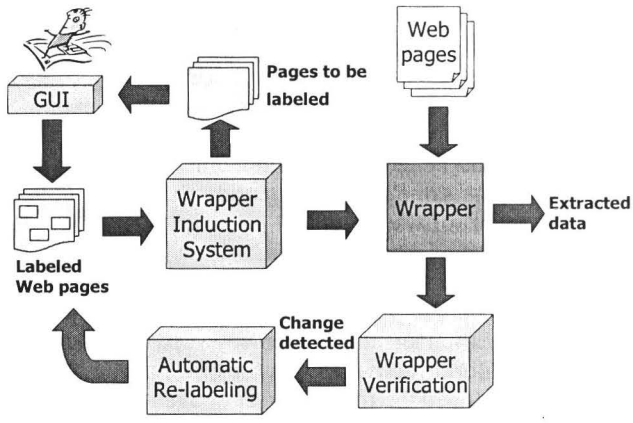

Figure 2: Wrapper induction, verification, and reinduction process

pair wrappers by learning a description of the content extracted by a wrapper [Lerman *et al.*, 2003]. This approach learns a pattern by using a hierarchy of pattern types, such as number or capitalized word, and then learning a description of the beginning and ending of the information that is being extracted. The resulting description or learned patterns are then stored and compared to the information being extracted. The patterns are compared statistically to avoid false positives due to examples that have not been seen before. In a large test set, this approach was able to identify 95% of the Web sites that had changed.

Once the system has identified a source that has changed, the learned patterns can then be used to automatically relabel the site and run the labeled examples through Stalker, the wrapper learning system. The wrapper learning, validation, and reinduction process are illustrated in Figure 2.

4 Linking Information

Once data has been extracted from a web site, then one frequently needs to combine it with other information in order to perform some task. For example, if one wants to build an agent for recommending nearby restaurants, then one might want to combine the data on a restaurant review site with the department of health site to ensure that the restaurant has an adequate health rating. The problem is that the information on these two distinct sites will often refer to the restaurants in different ways – the name, address, and phone number may all have slight variations that preclude simply joining the two sites across a single attribute.

To address this problem, we developed a machine learning approach to record linkage [Tejada *et al.*, 2002; 2001]. In this approach the system, called ActiveAtlas, compares the attributes that are shared across the two data sets and learns two things. First, it uses a committee of decision tree learners to decide whether two records are matched based on the strength of the match of the various attributes (Figure 3). Second, it improves the accuracy of these matches by learning an appropriate set of weights on a library of transformation rules (Figure 4). ActiveAtlas takes an active learning approach to

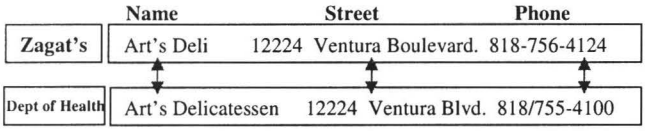

Figure 3: ActiveAtlas learns which attributes are important to decide whether two records should be linked

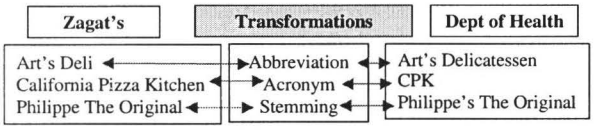

Figure 4: ActiveAtlas also learns the weighting of the transformation rules for an application

select examples for the user to label in order maximize the accuracy of the matches and minimize the amount of user input. Compared to other approaches to this problem [Cohen, 2000; Sarawagi and Bhamidipaty, 2002], the combination of the transformation weight learning and the active learning of the rules allows the system to achieve very high accuracy in matching entities.

5 Planning to Integrate Sources

Once an agent has access to the sources and can link the data across sources, the problem remains how to compose a set of information sources to perform some task. We developed an approach to automatically planning the integration of these sources to answer queries. In this approach, which is implemented in the Ariadne information mediator [Knoblock *et al.*, 1998; 2001], the contents of the sources available to the system are described using a common model [Ambite *et al.*, 2001]. The system uses this model to create a plan that specifies both the data sources and the specific operations to be performed to satisfy a request.

One of the interesting problems is that due to the large search space and the need to optimize the plans, traditional planning techniques do not scale. To overcome this problem we developed a general-purpose planning approach, called Planning by Rewriting [Ambite and Knoblock, 2001], and we use it as the planner for Ariadne [Ambite and Knoblock, 2000]. In Planning by Rewriting, the system starts with an initial, but suboptimal plan, and then the planner performs a local search through the space of plan transformations to maximize the given evaluation criterion. In the query planning application, the planner searches through the space of sources and the operations on these sources to find an efficient way to process a query. Figure 5 shows a simple example of how Planning by Rewriting searches through the space of plan transformations. Researchers have explored a variety of approaches to this general problem of planning for information gathering (see [Lambrecht and Kambhampati, 1997] for a summary of this work).

We are currently exploring planning techniques for composing Web Services [Thakkar *et al.*, 2003]. The Web Service infrastructure provides access to online sources in a form with which agents can interact without having to construct a wrapper for a site. In addition, the Semantic Web provides semantic-level descriptions of the services that are available [Ankolenkar *et al.*, 2002]. Our approach to integration planning builds on previous work on data integration [Levy, 2000] and applies the inverse rules approach of Duschka [1997].

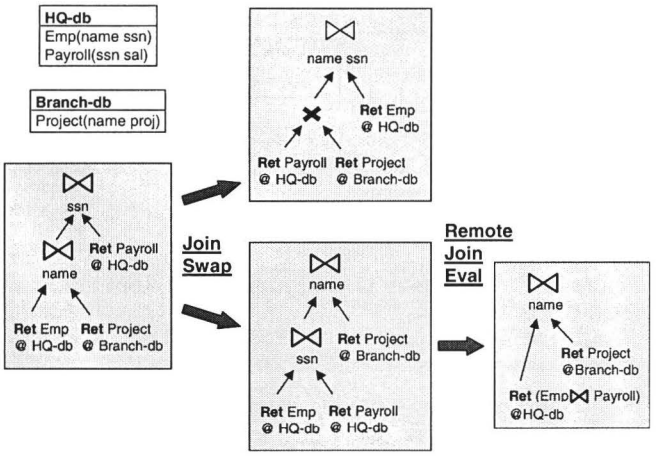

Figure 5: Planning by Rewriting searches through the space of possible transformations on a plan

One challenge is that the sources available online often have restrictions on how they can be accessed in that certain inputs may be required to access a source. In the inverse rules framework this means that computing a complete set of answers to a query may require recursion. Since it can be expensive to execute integration plans in a language such as Datalog, we have developed an approach to automatically convert the recursive plans produced by the inverse rules algorithm into a streaming dataflow execution framework that can efficiently execute these plans [Thakkar and Knoblock, 2003]. This execution framework is described next.

6 Executing Plans

Given a plan for performing some task on the Web, an agent needs to be able to efficiently execute this plan. In the Web environment, sources can be quite slow and the latencies of the sources are also unpredictable since they can be caused by heavy loads on both servers and networks. Since the primary bottleneck of most agent plans on the web is retrieving data from online sources, we would like to execute information requests as early as possible. To address these issues, we have developed a streaming dataflow language and executor, called Theseus [Barish and Knoblock, 2002a], which is optimized for the Web environment in the following three ways. First, since the executor is based on a dataflow paradigm, actions are executed as soon as the data becomes available. Second, Theseus performs the actions in a plan in separate threads, so they can be run asynchronously and in parallel. Third, the system streams the output from one action to the next so that sequential operations can be executed in parallel.

Theseus is similar to network query engines, such as Telegraph [Hellerstein *et al.*, 2000] or Tukwila [Ives *et al.*, 2002],

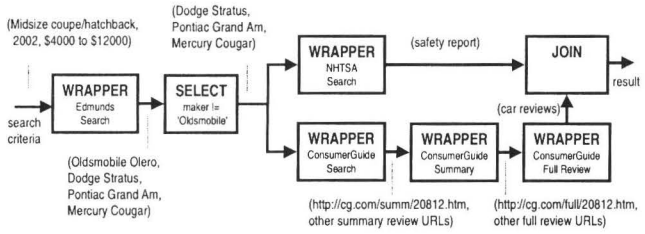

Figure 6: Example plan for integrating data from three car-related sites

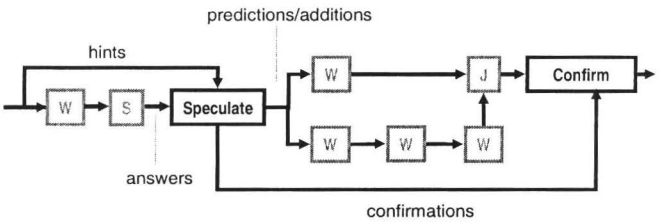

Figure 7: Augmented plan for speculative execution

in that they are also streaming dataflow execution systems. However, the network query engines focus on the efficient execution of of XML queries, while Thesues provides an expressive language for expressing information gathering and monitoring plans. The Theseus language supports capabilities that go beyond network query engines in that it supports recursion, notification operations, and writing and reading from databases to support monitoring tasks.

Recently we developed an approach to increase the potential parallelism in a streaming dataflow execution system. This optimization technique, called speculative execution [Barish and Knoblock, 2002b; 2003], attempts to predict the results of an operation based on data and patterns that it has seen in the past. The predicted results can then be used to speculate about the operations that will need to be performed later in the plan. The system decides where to speculate by analyzing a plan and determining the critical paths. On these paths it then inserts a "speculate" operation, which uses input to earlier operations to predict the input to later operations. The system also inserts a "confirm" operation, which ensures that the final result is correct regardless of whether the prediction is correct. This approach to optimizing streaming dataflow plans can achieve arbitrary speedups by speculating on the speculations. If the system is able to make accurate predictions, the executor could speculate on all of the input, execute the entire plan in parallel, and then confirm all of the results.

Figure 6 shows an example agent plan for integrating car-related data from three online sources. This plan first uses the Edmunds.com site to find the midsize cars priced between $4000 and $12000. Next it selects out those cars made by Oldsmobile. Then for each of those cars, in parallel it calls both the NHTSA site to get safety reports and the Consumer Guide site to retrieve car reviews. Finally, all of this information is combined into a single report. Figure 7 shows an abstract version of the same plan with the speculation operations inserted into the plan. The use of speculative execution in this plan makes it possible to invoke all three web sites in parallel.

The effectiveness of the speculation technique depends on making accurate predictions. We have developed a learning system that uses decision tree learning to make predictions on similar inputs and transducer learning to discover patterns in Web navigation paths. The learning system is described in more detail in [Barish and Knoblock, 2003].

7 Future Directions

Given the growing interest in both Web Services and the Semantic Web [Hendler, 2001], we believe it will become easier to rapidly create and deploy agents on the Web. As more semantic information becomes available about online sources, we plan to exploit this information to automatically discover and integrate new sources of information. Similarly, as web services become available that can support transactions, we plan to move beyond gathering and monitoring of online sources to build agents that perform more complex tasks, such as not just finding the lowest price ticket, but also purchasing it.

There are many possible uses of agents for retrieving and monitoring data from online sources. Ideally, users could define their own tasks that they want an agent to perform. For example, I might want an agent that monitors airfares and notifies me the moment I can buy a ticket to Hawaii for less then $300. Someone else might want an agent to monitor for travel delays in their connecting airport and notify them when the average delay exceeds 30 minutes. The possibilities are endless. We are currently working on what we call an Agent Wizard, which allows the user to define new agents for monitoring tasks simply by answering a set of questions about the task. The resulting system will work similar to the Excel Chart Wizard, which converts numerical data into charts by asking the user a set of questions. The Agent Wizard will automatically build the corresponding Theseus plan and schedule the monitoring task for the user.

Another exciting direction is to deploy agents to collect and learn about online data and then use the results to make predictions about the world. For example, we recently developed a system called Hamlet that advises a user about whether they should immediately buy a ticket on a particular flight or wait for a possible drop in the price [Etzioni *et al.*, 2003]. Hamlet makes these recommendations by collecting data on the current pricing of flights on a particular route and then learning a model of the pricing in order to make predictions about future price behavior. In a simulation using real data, Hamlet was able to save 607 simulated passengers $283,904, which was 88.6% of the savings possible with complete knowledge of the future price changes. Hamlet provides a compelling example of the potential of information agents.

8 Conclusion

The World Wide Web provides a tremendous opportunity for AI researchers to build, deploy and test software agents. The Web provides a real world environment that sidesteps many

of the difficult issues of sensing and control and makes it possible to explore higher level capabilities. We have developed the tools and infrastructure for rapidly constructing agents on the Web for performing various types of information gathering and integration tasks.

While the agents we have developed are extremely useful, there are many interesting and challenging problems that remain to be solved in order to widely deploy agents on the Web. Agents need to be able to robustly accomplish their tasks, responding appropriately to failures of various types. They must be able to communicate flexibly with people and other software agents, ideally in natural language. They need the ability to explain their behavior, especially as the tasks they perform become more complex. And, of course, we want agents that can learn from their past experience to both broaden their capabilities and improve their performance.

Acknowledgments

I want to thank my collaborators for their many contributions to the projects described in this paper. Steve Minton has worked closely with me on defining, building, and executing many of the research projects described here. Jose Luis Ambite, Greg Barish, Maria Muslea, Jean Oh, Snehal Thakkar, and Rattapoon Tuchinda all helped build the travel application of the Electric Elves. Ion Muslea developed the wrapper learning systems, Kristina Lerman developed the wrapper maintenance and repair techniques, Sheila Tejada developed the record linkage approach, Snehal Thakkar and Jose Luis Ambite developed the various query planning algorithms, and Greg Barish built the Theseus executor and speculative execution techniques. I also want to thank both Doug Dyer and Robert Herklotz for their support of my research over the years.

This research was supported in part by the Air Force Office of Scientific Research under grant numbers F49620-01-1-0053 and F49620-02-1-0270, in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under contract/agreement numbers F30602-01-C-0197 and F30602-00-1-0504, in part by the United States Air Force under contract number F49620-02-C-0103, and in part by a gift from the Microsoft Corporation.

The U.S.Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copy right annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

References

[Ambite and Knoblock, 2000] Jose Luis Ambite and Craig A. Knoblock. Flexible and scalable cost-based query planning in mediators: A transformational approach. *Artificial Intelligence Journal*, 118(1-2):115–161, April 2000.

[Ambite and Knoblock, 2001] Jose Luis Ambite and Craig A. Knoblock. Planning by rewriting. *Journal of Artificial Intelligence Research*, 15:207–261, 2001.

[Ambite *et al.*, 2001] Jose Luis Ambite, Craig A. Knoblock, Ion Muslea, and Andrew Philpot. Compiling source descriptions for efficient and flexible information integration. *Journal of Intelligent Information Systems*, 16(2):149–187, March 2001.

[Ambite *et al.*, 2002] Jose Luis Ambite, Greg Barish, Craig A. Knoblock, Maria Muslea, Jean Oh, and Steven Minton. Getting from here to there: Interactive planning and agent execution for optimizing travel. In *Proceedings of the Fourteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-2002)*, pages 862–869, AAAI Press, Menlo Park, CA, 2002.

[Ankolenkar *et al.*, 2002] Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry R. Payne, and Katia Sycara. Daml-s: Web service description for the semantic web. In *Proceedings of the First International Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002.

[Barish and Knoblock, 2002a] Greg Barish and Craig A. Knoblock. An efficient and expressive language for information gathering on the web. In *Proceedings of the AIPS-2002 Workshop on Is there life after operator sequencing? – Exploring real world planning*, pages 5–12, Tolouse, France, 2002.

[Barish and Knoblock, 2002b] Greg Barish and Craig A. Knoblock. Speculative execution for information gathering plans. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, pages 184–193, AAAI Press, Menlo Park, CA, 2002.

[Barish and Knoblock, 2003] Greg Barish and Craig A. Knoblock. Learning value predictors for the speculative execution of information gathering plans. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, Acapulco, Mexico, 2003.

[Barish *et al.*, 2000] Greg Barish, Craig A. Knoblock, Yi-Shin Chen, Steven Minton, Andrew Philpot, and Cyrus Shahabi. The TheaterLoc virtual application. In *Proceedings of Twelfth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, Austin, Texas, 2000.

[Chalupsky *et al.*, 2001] Hans Chalupsky, Yolanda Gil, Craig A. Knoblock, Kristina Lerman, Jean Oh, David V. Pynadath, Thomas A. Russ, and Milind Tambe. Electric elves: Applying agent technology to support human organizations. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, 2001.

[Cohen, 2000] William Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18:288–321, 2000.

[Crescenzi *et al.*, 2001] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.

[Decker *et al.*, 1997] Keith Decker, Anandeep Pannu, Katia Sycara, and Mike Williamson. Designing behaviors for information agents. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 404–412, ACM Press, New York, 1997.

[Duschka, 1997] Oliver M. Duschka. *Query Planning and Optimization in Information Integration*. PhD thesis, Stanford University, Department of Computer Science, 1997.

[Etzioni and Weld, 1994] Oren Etzioni and Daniel S. Weld. A softbot-based interface to the Internet. *Communications of the ACM*, 37(7), 1994.

[Etzioni *et al.*, 2003] Oren Etzioni, Craig A. Knoblock, Rattapoom Tuchinda, and Alexander Yates. To buy or not to buy: Mining airline fare data to minimize ticket purchase price. Submitted for Publication, 2003.

[Etzioni, 1996] Oren Etzioni. Moving up the information food chain: Deploying softbots on the world wide web. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1322–1326, AAAI Press / MIT Press ,Menlo Park, 1996.

[Hellerstein *et al.*, 2000] Joseph M. Hellerstein, Michael J. Franklin, Sirish Chandrasekaran, Amol Deshpande, Kris Hildrum, Sam Madden, Vijayshankar Raman, and Mehul A. Shah. Adaptive query processing: technology in evolution. *IEEE Data Engineering Bulletin*, 23(2):7–18, 2000.

[Hendler, 2001] James Hendler. Agents on the web. *IEEE Intelligent Systems, Special Issue on the Semantic Web*, 16(2):30–37, March/April 2001.

[Hong and Clark, 2001] Theodore W. Hong and Keith L. Clark. Using grammatical inference to automate information extraction from the Web. In *Principles of Data Mining and Knowledge Discovery, Lecture Notes in Computer Science*, volume 2168, pages 216–227. Springer-Verlag, 2001.

[Hsu and Dung, 1998] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[Ives *et al.*, 2002] Zachary G. Ives, Alon Y. Halevy, and Daniel S. Weld. An XML query engine for network-bound data. *VLDB Journal*, 11(4):380–402, 2002.

[Knoblock *et al.*, 1998] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Pragnesh Jay Modi, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998.

[Knoblock *et al.*, 2001] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. The ARIADNE approach to web-based information integration. *International Journal of Cooperative Information Systems (IJCIS), Special Issue on Intelligent Information Agents: Theory and Applications*, 10(1/2):145–169, 2001.

[Knoblock *et al.*, 2003] Craig A. Knoblock, Kristina Lerman, Steven Minton, and Ion Muslea. Accurately and reliably extracting data from the web: A machine learning approach. In Piotr S. Szczepaniak, Javier Segovia, Janusz Kacprzyk, and Lotfi A. Zadeh, editors, *Intelligent Exploration of the Web*, pages 275–287. Springer-Verlag, Berkeley, CA, 2003.

[Krulwich, 1996] Bruce Krulwich. The bargainfinder agent: Comparison price shopping on the internet. In *Agents, Bots, and other Internet Beasties*, pages 257–263. Macmillan Publishing, May 1996.

[Kushmerick, 1997] Nicholas Kushmerick. *Wrapper Induction for Information Extraction*. PhD thesis, Department of Computer Science and Engineering, University of Washington, 1997.

[Kushmerick, 2000] Nicholas Kushmerick. Wrapper verification. *World Wide Web*, 3(2):79–94, 2000.

[Lambrecht and Kambhampati, 1997] Eric Lambrecht and Subbarao Kambhampati. Planning for information gathering: A tutorial survey. ASU CSE technical report 96-017, Department of Computer Science and Engineering, Arizona State University, May 1997.

[Lerman *et al.*, 2001] Kristina Lerman, Craig A. Knoblock, and Steven Minton. Automatic data extraction from lists and tables in web sources. In *Proceedings of the IJCAI 2001 Workshop on Adaptive Text Extraction and Mining*, Seattle, WA, 2001.

[Lerman *et al.*, 2003] Kristina Lerman, Steven N. Minton, and Craig A. Knoblock. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, 18:149–181, 2003.

[Levy and Weld, 2000] Alon Y. Levy and Daniel S. Weld. Intelligent internet systems. *Artificial Intelligence*, 118(1-2):1–14, April 2000.

[Levy, 2000] Alon Y. Levy. Logic-based techniques in data integration. In Jack Minker, editor, *Logic-Based Artificial Intelligence*. Kluwer Academic Publisher, 2000.

[Muslea *et al.*, 2000] Ion Muslea, Steven Minton, and Craig A. Knoblock. Selective sampling with redundant views. In *Proceedings of the 17th National Conference on Artificial Intelligence*, 2000.

[Muslea *et al.*, 2001] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2), March 2001.

[Muslea *et al.*, 2003] Ion Muslea, Steven Minton, and Craig A. Knoblock. Active learning with strong and weak views: A case study on wrapper induction. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, Acapulco, Mexico, 2003.

[Muslea, 2002] Ion Muslea. *Active Learning with Multiple Views*. PhD thesis, Department of Computer Science, University of Southern California, 2002.

[Perkowitz *et al.*, 1997] Mike Perkowitz, Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. Learning to understand information on the internet: An example-based approach. *Journal of Intelligent Information Systems*, 8(2):133–153, 1997.

[Sarawagi and Bhamidipaty, 2002] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, Canada, 2002.

[Tejada *et al.*, 2001] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8), 2001.

[Tejada *et al.*, 2002] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, Canada, 2002.

[Thakkar and Knoblock, 2003] Snehal Thakkar and Craig A. Knoblock. Efficient execution of recursive integration plans. In *Proceedings of 2003 IJCAI Workshop on Information Integration on the Web*, Acapulco, Mexico, 2003.

[Thakkar *et al.*, 2003] Snehal Thakkar, Jose-Luis Ambite, and Craig A. Knoblock. A view integration approach to dynamic composition of web services. In *Proceedings of 2003 ICAPS Workshop on Planning for Web Services*, Trento, Italy, 2003.