ACADEMIA: An Agent-Maintained Database based on Information Extraction from Web Documents

Mario Magnanelli and **Antonia Erni** and **Moira Norrie**
Institute for Information Systems
Swiss Federal Institute of Technology (ETH)
ETH-Zentrum, CH-8092 Zurich, Switzerland.
{magnanel,norrie,erni}@inf.ethz.ch

Abstract

We describe an Internet agent which gathers information from the Web in order to maintain a local database and ensure its currency. As a specific application, we detail an agent maintaining a database with information about academic contacts, their projects and publications. Agent operation is driven by an extraction profile which specifies what and how information is to be extracted from Web documents. The agent detects new and updated information and, when the confidence level is above a user-specified threshold, automatically updates the database accordingly.

1 Introduction

The World Wide Web (WWW) has become a major source of information about all areas of interest. Users typically spend many hours searching not only for new Web documents, but also for updates to documents. For example, an academic may look for new technical reports, a financial analyst for new economic data and a computer enthusiast for new software products and versions. Further, it also requires significant time to download information and effort to organize it in a convenient form.

To assist users in the tasks of finding, fetching and working with information published in Web documents, we use an Internet agent to gather information and store it in a local client database, thereby allowing users to browse, query and process that information at their convenience. Agent operation is driven by a combination of an extraction profile specifying what and how information is to be extracted from Web documents and the local database specifying the particular entities of interest. Thus, the user accesses the local database system and it is the responsibility of the agent to maintain this database and ensure its currency.

While the approach is general and the agent dynamically configurable, here we use a specific application system, ACADEMIA, to describe the operation of the agent and the information extraction process. ACADEMIA is a system to support academics by automatically keeping track of contact information for other researchers – such as telephone numbers and email addresses – and also information on their projects and publications.

The ACADEMIA agent runs in the background, periodically searching the Web. The frequency of the search is specified by the user. By creating an entry for each researcher of interest, the user effectively specifies the domain of interest and the agent uses this information to know who or what to search for.

The information extraction process is controlled by an *extraction profile* which specifies *how* information is to be extracted from Web documents based on a combination of keyword searches, term matching and proximity measures. Confidence measures are associated with the various extraction patterns, thereby allowing the agent to calculate reliability scores for extracted information items. These reliability scores, along with user-specified confidence thresholds, determine whether, for a given information item, the agent updates the database directly or consults the user.

ACADEMIA combines techniques developed in various research areas for extracting information from Web documents. In the database area, systems are being developed to allow querying over dynamically generated Web documents. For example, in [Hammer *et al.*, 1997], a language is proposed for specifying extraction patterns to enable structured objects to be constructed from information contained in HTML documents. These systems only work over fixed Web sites for which patterns have been specified. In contrast, our agent does not base extraction on fixed patterns and can extract information from any form of Web page.

Our agent does use pattern-based extraction mechanisms to extract information on publications and projects. However, the agent itself generates these patterns based on the structure of individual items found in repeating items such as HTML lists and tables. Similar techniques have been used, for example, in comparative shopping agents to extract information from specific sites of on-line stores [Doorenbos *et al.*, 1997]. However, these

agents use training keywords to learn the patterns of announced pages, while our agent finds pages by itself and does not need explicit training keywords.

Work such as [Menczer, 1997] and [Armstrong *et al.*, 1995] use more complex retrieval functions, but focus mainly on presenting whole Web pages to the user. In our agent, the extraction profile drives retrieval by specifying how to find possible pages of interest and its main task is to then extract information from these pages.

Section 2 describes the components and operation of the ACADEMIA system and section 3 gives details of the extraction profile and the extraction process. Section 4 describes the specific process of extracting information on publications. Section 5 describes how confidence values are assigned to extracted facts. Finally, concluding remarks are given in section 6.

2 ACADEMIA System

ACADEMIA is used to reduce the work of an academic in finding and updating information about other researchers. While we use this specific application to explain our general extraction mechanisms, we note that the general concepts of this system may be used in other applications and, with this aim in mind, the agent can be dynamically configured. Figure 1 shows the components of the ACADEMIA system and the work flow between them.

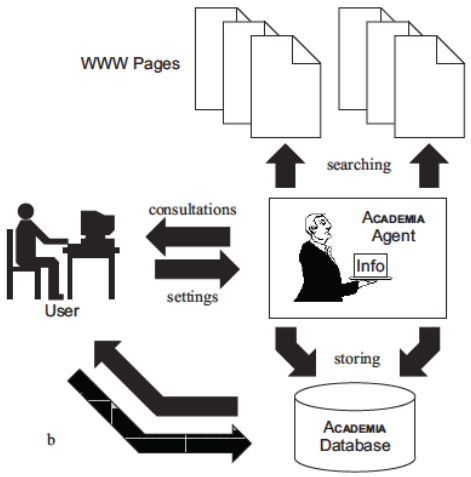

Figure 1: The components of ACADEMIA

The ACADEMIA database is implemented using the OMS object-oriented database management system (DBMS) described in [Norrie and Würgler, 1997; Norrie, 1993]. OMS provides a graphical browser, full query language and methods which are used to support user operations such as downloading documents. Since the system also supports URLs as a base type, viewing Web pages and sending email via an Internet browser can be done directly from OMS. Further, since a generic WWW interface for OMS is available, the ACADEMIA database can also be accessed through such a browser.

The key contact information in the database consists of person names and WWW addresses. The name is necessary to identify the person, while the address is a general starting point for the agent to search for updates.

The database also stores general facts about persons such as title, address, photo and information about research activities including the titles of publications, URLs leading to abstracts or a publication file, project titles and URLs of pages containing further information on the project.

The user accesses the database directly to retrieve and process information on academic contacts. The ACADEMIA agent provides a *value-added* service by using information extracted from Web documents to maintain the database and ensure its currency. The agent may either update the database directly, or consult with the user as to whether or not it should perform the updates.

The extraction process of the agent is specified by an extraction profile. For a given application system such as ACADEMIA, this profile is provided as part of the system. However, the user could adapt it to search for additional information. In section 3, the profile is explained in detail.

An ACADEMIA agent runs in the background according to the periodicity specified by the user. It first reads the name and WWW-address of each person in the database to determine the search domain. If the agent does not find a WWW-address for a person, it tries to find a WWW-address by using the AltaVista search engine. In this case, the only search arguments are the first and last name of the person and, of course, it is not sure whether relevant documents will be found. The agent performs a search with each of the first ten pages returned by AltaVista and, in the case that information is found, later consults with the user who decides whether this information is reliable or not and should be stored in the database. We note that other search engines – including those specifically for personal home pages – have been tried and we are investigating which combinations of search engines are best for our application.

Given one or more possible home pages for a person, the agent starts to extract information from these and referenced pages. Searching home pages is done in two basic ways – keyword-based and pattern-based search. In the case of keyword-based search, the agent searches for keywords as specified in the extraction profile. For each keyword, a set of options is specified which tells the agent what information may be found in proximity to the keyword. For example, if a URL follows the keyword "www", it is likely to be a link to another home page. Details of the extraction process and the format of the extraction profile are given in the next section. Although such keyword searching is

relatively simple, it has proved effective and is used in ACADEMIA to find general information about a person and also potential links to pages containing publication lists or project descriptions.

Pattern-based search is used to find information about publications and projects. In most cases, this information is represented in lists and cannot be extracted by the keyword approach. For example, publications are frequently represented within Web documents as an HTML list with each item giving the authors, title, publication information and one or more URLs to download the document. The keywords "author" or "title" do not occur explicitly. Our agent therefore tries to detect a recurring pattern in the HTML page indicating the occurrence of such a list. This is based on HTML-commands around text items and the use of lists, tables and different fonts to structure information. Details of pattern-based search is given in section 4.

As mentioned, the agent starts searching from a potential home page and repeatedly follows interesting links to search for further information. Links are collected in a search list and can be of three types: links that likely lead to general information, to publications or to research topics. The agent searches each link using the corresponding search technique defined for each type of link. A future version of ACADEMIA will contain the possibility for the user to alter these search techniques.

After the search for one person, a confidence value (CV) is computed for each piece of information found based on the reliability of the extraction pattern used to find that item and the level of corroborating and/or contradictory information found. For example, if the same telephone number is found in several places, the level of confidence will be high. However, if different phone numbers are found on different pages, the confidence will be low. These CVs can only be calculated at the end of the search since it is not possible to predict when and where items will be found.

Once the search is complete, the agent starts the interaction with the database. For every fact that has a CV greater than the user-specified threshold, the agent writes the fact in the database and records this action in a log which the user may access to examine the agent's actions. For facts which have CVs below the threshold, the agent will later consult the user who decides whether the fact will be stored or not. The agent stores the decisions of the user for future reference, thereby avoiding repeatedly asking the user the same questions. Whenever the user gains more confidence in the agent, he may reduce the threshold to give the agent greater autonomy.

3 Extraction Profile

In this section, we describe the extraction profile in detail. To assist understanding, we explain some

The profile consists of a set of extraction patterns each of which specifies a *keyword* to be searched for and also its significance in terms of the form of information to be extracted, the proximity of this information, any supporting keywords and an associated CV.

The general idea behind the extraction process based on these patterns is as follows. The agent first searches the page for a keyword of an extraction pattern. If a keyword is found, it indicates that information we seek may be located in the vicinity. There are several ways in which this information can be found. For example, consider the case of looking for a person's phone number. The keyword "phone" indicates that a phone number may follow. Phone numbers usually consist of digits with a few special characters. Further, it is very likely that this number follows immediately after the keyword. With this background knowledge, it is not difficult to extract a string that is likely to be the phone number – if it exists. As another example, consider finding the title of a person. If the keyword "prof" is found followed by the name of the person, it is likely to be their title.

Such reasoning leads to the specification of the various extraction patterns. At any stage, the user can add new patterns or refine existing ones. Part of the extraction profile for the ACADEMIA agent is shown in table 1.

Keyword	R	in	D	N	FN	ML	XL	C	Obj
prof	b	x	0	10	0	0	0	50	title
prof	b	x	0	24	10	0	0	100	title
publication	p	l	10	0	0	0	0	100	-

Table 1: Example 1 of the extraction profile

Each line in the profile corresponds to an extraction pattern specifying a keyword along with additional information to determine if a fact of the appropriate form has been found and how to extract that fact. The extraction pattern is specified in terms of a number of attributes – some of which are optional. We start by explaining the attributes shown in table 1.

Attribute R specifies the type of information to be extracted. The agent distinguishes between two main categories of information – reference and textual information. Textual information consists of facts to be extracted. It may be of several types. If `R=s`, a string value is to be extracted. If `R=b`, a Boolean value is returned indicating whether or not a specific term has been located. For example, for the title of a person, we simply want to know whether a specific designation such as "prof" appears in front of that person's name. Other types include email-addresses (`R=e`), dates (`d`) and images (`i`).

Reference information consists of links to other

general page (`l`), publication page (`p`) or research page (`r`), or it can be a link to a Web page where a "finger"-command is performed (`f`).

The next attribute, *in*, determines the position of the keyword in the Web document. The keyword may be found in usual text (`x`), in text belonging to a link (`k`), in the title of a page (`t`), in a header (`h`), inside of an HTML-command (`c`) or in a link reference (`l`). Thus, in row 3 of table 1, we specify that "publication" has to be found in a link – indicating that a reference to a Web page containing a list of publications has possibly been found.

D determines the locality of the information to be extracted in terms of the maximum distance (in characters) from the keyword. 0 means the result can be in any distance from the keyword.

The attributes *N* and *FN* can be used to specify that the surname or forename of the person must appear in proximity to the keyword. For example, the first extraction pattern of table 1 specifies that the surname of the person must appear at most 10 characters from the keyword "prof". This is used to check that the designation belongs to the person whose information we are seeking. The second line specifies that the forename also occurs. A 0 for either of these attributes means that the corresponding name does not have to occur. *ML* and *XL* determine the minimum, respectively maximum, length of the resulting information for types string and email-address.

The CV associated with an extraction pattern is specified in attribute *C*. It is given in percent. More about CVs is given in section 5.

The last attribute, *Obj*, is used to tell the agent where the extracted information is to be stored in the database. In the case of reference information, no information is stored and therefore *Obj* is unspecified.

In table 2, we show other optional attributes for specifying in more detail the format of values to be extracted. Note that we have omitted here the CVs which happen to all be 100%.

Keyword	R	in	D	CharSet	ML	XL	Obj
email	e	x	0	-	8	40	email
phone	s	x	6	+()0123456789/	8	22	phone
finger	f	k	0	-	0	0	finger

Table 2: Example 2 of the extraction profile

CharSet specifies all possible characters allowed to occur in a string value. These are used in table 2 to specify expected forms of telephone numbers. Thus, to find a phone number, the agent looks for a string containing only those characters and starting within a distance of 6 characters after the keyword "phone". The keyword must occur in usual text. The result is a phone number with a length between 8 and 22.

In the case of email and finger information, the extracted. Thus, for an email address, the agent automatically looks for a string containing a "@".

Another part of the extraction profile is given in table 3. *SK* is used to specify a second keyword that has to occur in the same Web document. *SKD* specifies the maximum distance of the second keyword from the first. 0 means any distance.

Keyword	R	in	D	SK	SKD	ML	XL	C	Obj
home	l	k	0	page	0	0	0	100	-
my	l	x	15	work	12	0	0	100	-
project	r	x	50	lead	-20	0	0	100	-

Table 3: Example 3 of the extraction profile

The first two extraction patterns in table 3 are used to get links to pages with general facts. The first specifies that keywords "home" and "page" must both occur in text that belongs to a link. The distance between them is not specified, but they have to occur in the same link text. The second line specifies that "work" should begin within 12 characters of "my" and both should appear in regular text. The extracted link to a further Web document of possible general interest has to be found within a maximum distance of 15 characters.

The third extraction pattern of table 3 is used to find a link to a page which may specifically contain information about projects. If keywords "project" and "lead" occur in usual text with "lead" appearing at most 20 characters before "project", a link within a distance of no more than 50 characters is assumed to be a possible link to a Web document listing projects. For example, a line of an HTML-page may contain the text: "Currently, I'm leading a project called Artemis". If, following Artemis, there is a link to the home page of this project, the extraction pattern would cause the agent to extract this link and search the resulting Web document for project information.

The specific values shown in the example tables were those which, during testing, led to good results. We chose them by analyzing the forms of many WWW-pages containing relevant information and then adapting the proximity values based on experience. More detailed information about the extraction profile and the keyword-based extraction process can be found in [Magnanelli, 1997].

4 Extraction of Publications

In this section, we describe pattern-based extraction by detailing how the agent extracts information about publications. We start by assuming that the agent has located a document (or part of a document) that is deemed likely to contain information on publications. The agent then looks for some form of pattern of repeated entries such as an HTML list or table structure. If the agent detects such a recurring pattern, it next tries to find

our agent operation, is ideal in terms of extracting information.

```
<H2>Object-Oriented Temporal Databases</H2>
<B>A. Steiner and M. C. Norrie.</B>
<I>Institute for Information Systems, ETH Zuerich.</I>
April 1997.<br>
Proc. 5th Int. Conf. on DASFAA 97, Melbourne, Australia
<br><br> Available files:
[<a href="ftp://.../97c-dasfaa.abstract">abstract</a>]
[<a href="ftp://.../97c-dasfaa.ps">postscript</a>]
<p>
<H2>New Programming Environment for Oberon</H2>
<B>J. Supcik and M. C. Norrie.</B>
<I>Institute for Information Systems, ETH Zuerich.</I>
March 1997.<br>
Proc. JMLC 97, Linz, Austria
<br><br> Available files:
[<a href="ftp://.../97b-jmlc.abstract">abstract</a>]
[<a href="ftp://.../97b-jmlc.ps">postscript</a>]
<p>
```

Figure 2: Part of an HTML publication list

Both entries shown contain the same structure of HTML-commands. We note that this case occurs seldomly as it may be that not all entries contain the same fields. For example, a particular publication entry may contain no date or proceedings. In fact, typically, the larger a pattern is, the more likely it is that there are small differences between several entries and our agent respects that.

Because of possible irregularities in items, we decided not to use every HTML-command to define the pattern of an entry. The tag "
", for example, never stands for a significant separation of two parts in an entry. Also the links beginning with "<a ...>' should not be used because not all publications may have referenced pages or postscript versions.

The agent first looks for the position of the name of the person in question. For example, in figure 2, we might look for publications of which Supcik is an author. "Supcik" is found between the HTML-tags <B> and </B>. The agent therefore assumes that for every item, the names of the authors will be located in the corresponding part.

The agent next tries to extract the title of the publication. For this, we used the observation that the title occurs towards the beginning of an item – either in the first or second position. Also, usually, the title contains at least twenty characters and seldomly contains commas. The agent uses these statistics to extract the titles from the entries. The associated CV will reflect the reliability that an extracted string is the title based on whether or not these various observations occur. Thus, if a title is less than twenty characters in length, it may still be extracted, but have a lower CV.

The agent also examines every link given in an entry. These links are also stored if they appear to

Pattern-based extraction is also used to look for information on research projects. However, in this case, the ability of the agent to extract information is not as good as for publications. The main reason for this is that information given about research projects tends to be less well-structured. In fact, it is often given as free text, without any heading or page name to indicate that it is indeed information about research topics or projects.

5 Confidence Values

Having explained how information is extracted from Web documents, we now describe how the agent determines the reliability of this information. As stated previously, each extraction pattern has an associated CV which gives a measure of the reliability of an extracted information item in isolation. To compute an overall confidence measure, the agent must consider the context in which the information was found and also the occurrence of any corroborating or conflicting information.

We refer to the CVs associated with extraction patterns as conditional possibilities, i.e.

> $C(f|k)$ = the possibility that fact f occurs given a keyword k

The idea of CVs is adapted from certainty factors as defined in [Buchanan and Shortliffe, 1984]. The main difference between those values is the range. Certainty factors normally range from -1 (complete uncertainty) to +1 (complete certainty). Our CVs range from 0 to infinity, because there exists no complete certainty whether a fact found is reliable. We let the user set a threshold which indicates the CV that a fact has to reach in order to seem reliable to the user.

Mathematically, there is no complete certainty but, in practice, we found many patterns which always led to reliable facts. Therefore, we decided to use percentage values for the CVs to indicate the reliability of the extraction pattern in terms of the number of cases in which the pattern leads to correct facts. Thus, a value of 50 percent means that only in half of all cases a fact extracted using that pattern is, in isolation, considered reliable.

It is however not sufficient to consider only the effectiveness of a given extraction pattern in calculating the CV of a fact. For example, it may indeed be the case that an extraction pattern leads correctly to a phone number, but that we have a low confidence that the page being analyzed contains information about the person in question. Thus, the CV of a fact also depends on the CV associated with the context. Each page is therefore assigned a CV that indicates how likely it is that facts on this page belong to the processed person.

> $C(p)$ = possibility that page p contains useful information on the focused person

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.