Learning Information Extraction Rules for Web Data Mining

Chia-Hui Chang
National Central University, Taiwan

Chun-Nan Hsu
Institute of Information Science, Academia Sinica, Taiwan

INTRODUCTION

The explosive growth and popularity of the World Wide Web has resulted in a huge number of information sources on the Internet. However, due to the heterogeneity and the lack of structure of Web information sources, access to this huge collection of information has been limited to browsing and keyword searching. Sophisticated Web-mining applications, such as comparison shopping, require expensive maintenance costs to deal with different data formats. The problem in translating the contents of input documents into structured data is called information extraction (IE). Unlike information retrieval (IR), which concerns how to identify relevant documents from a document collection, IE produces structured data ready for post-processing, which is crucial to many applications of Web mining and search tools.

Formally, an information extraction task is defined by its input and its extraction target. The input can be unstructured documents like free text that are written in natural language or semi-structured documents that are pervasive on the Web, such as tables, itemized and enumerated lists, and so forth. The extraction target of an IE task can be a relation of k-tuple (where k is the number of attributes in a record), or it can be a complex object with hierarchically organized data. For some IE tasks, an attribute may have zero (missing) or multiple instantiations in a record. The difficulty of an IE task can be complicated further when various permutations of attributes or typographical errors occur in the input documents.

Programs that perform the task of information extraction are referred to as *extractors* or *wrappers*. A wrapper is originally defined as a component in an information integration system that aims at providing a single uniform query interface to access multiple information sources. In an information integration system, a wrapper is generally a program that wraps an information source (e.g., a database server or a Web server) such that the information integration system can access that information source without changing its core query answering mechanism. In the case where the information source is a Web server, a wrapper must perform information extraction in order to extract the contents in HTML documents.

Wrapper induction (WI) systems are software tools that are designed to generate wrappers. A wrapper usually performs a pattern-matching procedure (e.g., a form of finite-state machines), which relies on a set of extraction rules. Tailoring a WI system to a new requirement is a task that varies in scale, depending on the text type, domain, and scenario. To maximize reusability and minimize maintenance cost, designing a trainable WI system has been an important topic in research fields, including message understanding, machine learning, pattern mining, and so forth. The task of Web IE differs largely from traditional IE tasks in that traditional IE aims at extracting data from totally unstructured free texts that are written in natural language. In contrast, Web IE processes online documents that are semi-structured and usually generated automatically by a server-side application program. As a result, traditional IE usually take advantage of natural language processing techniques such as lexicons and grammars, while Web IE usually applies machine learning and pattern-mining techniques to exploit the syntactical patterns or to lay out structures of the template-based documents.

BACKGROUND

In the past few years, many approaches of WI systems and how to apply machine learning and pattern mining techniques to train WI systems have been proposed with various degrees of automation. Kushmerick and Thomas (2003) conducted a survey that categorizes WI systems based on the wrapper programs' underlying formalism (whether they are finite-state approaches or Prolog-like logic programming systems). Sarawagi (2002) further distinguished deterministic finite-state approaches from probabilistic hidden Markov models in her 2002 VLDB tutorial. Another survey can be found in Laender, et al. (2002), which categorizes Web extraction tools into six classes based on their underlying techniques—declara-

tive languages, HTML structure analysis, natural language processing, machine learning, data modeling, and ontology.

MAIN THRUST

We classify previous work in Web IE into three categories. The first category contains the systems that require users to possess programming expertise. This category of wrapper generation systems provides specialized languages or toolkits for wrapper construction, such as W4F (Sahuguet & Azavant, 2001) and XWrap (Liu et al., 2000). Such languages or toolkits were proposed as alternatives to general-purpose languages in order to allow programmers to concentrate on formulating the extraction rules without being concerned about the detailed process of input strings. To apply these systems, users must learn the language in order to write their extraction rules. Therefore, such systems also feature user-friendly interfaces for easy use of the toolkits. However, writing correct extraction rules requires significant programming expertise. In addition, since the structures of Web pages are not always obvious and change frequently, writing specialized extraction rules can be time-consuming, error-prone, and not scalable to a large number of Web sites. Therefore, there is a need for automatic wrapper induction that can generalize extraction rules for each distinct IE task.

The second category contains the WI systems that require users to label some extraction targets as training examples for WI systems to apply a machine-learning algorithm to learn extraction rules from the training examples. No programming is needed to configure these WI systems. Many IE tasks for Web mining belong to this category; for example, IE for semi-structured text such as RAPIER (Califf & Mooney, 1999), SRV (Freitag, 2000), WHISK (Soderland, 1999), and for IE for template-based pages such as WIEN (Kushmerick et al., 2000), SoftMealy (Hsu and Dung, 1998), STALKER (Muslea, et al., 2001), and so forth. Compared to the first category, these WI systems are preferable, since general users, instead of only programmers, can be trained to use these WI systems for wrapper construction.

However, since the learned rules only apply to Web pages from a particular Web site, labeling training examples can be laborious, especially when we need to extract contents from thousands of data sources. Therefore, researchers have focused on developing tools that can reduce labeling effort. For instance, Muslea, et al. (2002) proposed selective sampling, a form of active learning that reduces the number of training examples. Chidlovskii, et al. (2000) designed a wrapper generation system that requires a small amount (one training record) of labeling by the user. Earlier annotation-based WI systems place emphasize on the learning techniques in their paper. Recently, several works have been proposed to simplify the annotation process. For example, Lixto (Baumgartner et al., 2001), DEByE (Laender et al., 2002) and OLERA (Chang & Kuo, 2004) are three such systems that stress the importance of how annotation or examples are received from users. Note that OLERA also features the so-called semi-supervised approach, which receives rough rather than exact and perfect examples from users to reduce labeling effort.

The third category contains the WI systems that do not require any preprocessing of the input documents by the users. We call them *annotation-free WI systems*. Example systems include IEPAD (Chang & Lui, 2001), RoadRunner (Crescenzi et al., 2001), DeLa (Wang & Lochovsky, 2003), and EXALG (Arasu & Garcia-Molina, 2003). Since no extraction targets are specified, such WI systems make heuristic assumptions about the data to be extracted. For example, the first three systems assume the existence of multiple tuples to be extracted in one page; therefore, the approach is to discover repeated patterns in the input page. With such an assumption, IEPAD and DeLa only apply to Web pages that contain multiple data tuples. RoadRunner and EXALG, on the other hand, try to extract structured data by deducing the template and the schema of the whole page from multiple Web pages. Their assumption is that strings that are stationary across pages are presumably template, and strings that are variant are presumably schema and need to be extracted. However, as commercial Web pages often contain multiple topics where a lot of information is embedded in a page for navigation, decoration, and interaction purposes, their systems may extract both useful and useless information from a page. However, the criterion of what is useful is quite subjective and depends on the application. In summary, these approaches are, in fact, not fully automatic. Rather, post-processing is required for users to select useful data and to assign the data to a proper attribute.

Task Difficulties

A critical issue of WI systems is what types of documents and structuring variations can be handled. Documents can be classified into structured, semi-structured, and unstructured sets (Hsu & Dung, 1998). Early IE systems like RAPIER, SRV, and WHISK are designed to handle documents that contain semi-structured texts, while recent IE systems are designed mostly to handle documents that contain semi-structured data (Laender et al., 2002). In this survey, we focus on semi-structured data extraction and possible structure variation. These include missing data attributes, multi-valued attributes, attribute permutations, nested data structures, and so forth. Table 1 lists