Transcendence: Enabling a Personal View of the Deep Web

Jeffrey P. Bigham, Anna C. Cavender, Ryan S. Kaminsky, Craig M. Prince and Tyler S. Robison
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195 USA
{jbigham, cavender, rkamin, cmprince, trobison}@cs.washington.edu

ABSTRACT

A wealth of structured, publicly-available information exists in the *deep web* but is only accessible by querying web forms. As a result, users are restricted by the interfaces provided and lack a convenient mechanism to express novel and independent extractions and queries on the underlying data. *Transcendence* enables personalized access to the deep web by enabling users to partially reconstruct web databases in order to perform new types of queries. From just a few examples, Transcendence helps users produce a large number of values for form input fields by using unsupervised information extraction and collaborative filtering of user suggestions. Structural and semantic analysis of returned pages finds individual results and identifies relevant fields. Users may revise automated decisions, balancing the power of automation with the errors it can introduce. In a user evaluation, both programmers and non-programmers found Transcendence to be a powerful way to explore deep web resources and wanted to use it in the future.

ACM Classification H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.
General Terms Design, Human Factors, Algorithms

Author Keywords

Deep Web, Information Extraction, Web Forms

INTRODUCTION

The *deep web* consists of web information that is publicly available but accessible only by querying web forms. This prodigious resource of information is estimated to be 400 to 550 times larger than the *surface web* which is indexable by traditional search engines [9]. Information in the deep web is not easily crawled automatically, and, as a result, users are limited in how they can use this information by the provided interface. In this paper, we introduce *Transcendence*, a tool that enables users to issue multiple queries to online resources in parallel, merge the results and perform types of queries not supported by the original interface. The system leverages unsupervised information extraction and collaborative filtering to suggest relevant input values to web forms and automatically detects the structure of the results, enabling it to merge the output from multiple queries.

IUI'08, January 13-16, 2008, Maspalomas, Gran Canaria, Spain.

Online search tools are providing more features and flexibility to help users navigate the ever-increasing amount of data in deep web resources: organizations are providing web interfaces for their existing data and user communities are generating a wealth of new data. As an example, *kayak.com* makes the common task of finding airline tickets on the web easier through richer interfaces that enable users to specify flexible dates, acceptable nearby airports or a preference for direct flights. Because user interface designers cannot predict the needs of all users, the expressiveness of users' searches are still restricted by the provided interfaces. Transcendence provides a mechanism for users to easily specify a crawler that can collect relevant data from the deep web and expose it in a way that other web visualizers can consume. Transcendence users can search for flights on *kayak.com* with optional destinations that are not sufficiently near one another to be suggested by the existing interface and can search using dates that are weeks apart - neither of which is supported by the current interface.

As another example, consider Jane, who is a new student at the University of Washington looking for an apartment on

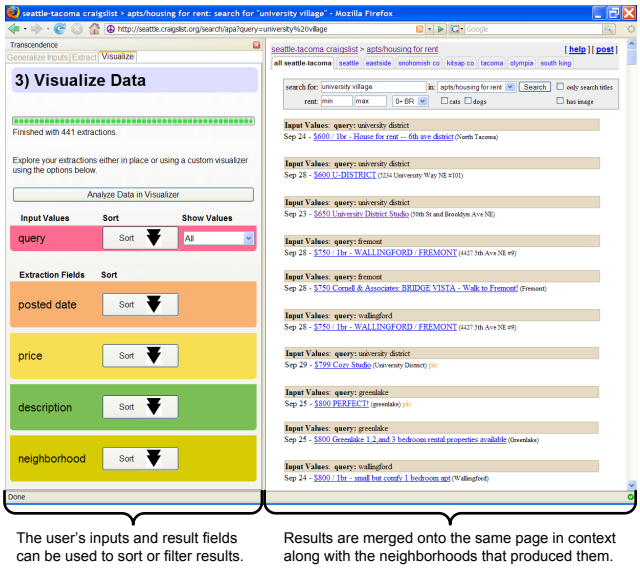

Figure 1. Craigslist results for Seattle neighborhoods sorted by price.

craigslist.org. Jane wants an apartment near the university, but when she arrives at the Craigslist page for Seattle, she is presented with numerous listings irrelevant to her, such as ads for apartments in cities and neighborhoods far from the university, such as in Renton or West Seattle. She knows that the University District and University Village neighborhoods are close to campus and decides to search using these terms. Jane first enters both of these values into a search field enhanced by Transcendence. Transcendence generalizes the provided phrases to find others that likely belong to the same implicit class - in this case, Wedgewood, Greenlake and Capitol Hill, which are neighborhoods close to campus. The system also returns several questionable suggestions, such as the University of Washington and the University District Street Fair, which are not neighborhoods. The former produces results that include University of Washington in them and therefore may be popular with students, while the latter produces no results and, therefore, has no end effect.

Transcendence issues queries for all of these terms in parallel and combines the results in context on a Craigslist results page (Figure 1). Transcendence has automatically identified fields appearing in the results that may be important to Jane, including the price of the apartment and the neighborhood where it is located, and provides an interface that enables her to sort and filter results based on these fields. Using the Transcendence data analyzer, Jane computes the average and median prices for apartments in these areas. Transcendence enabled her to find a nice studio in the Greenlake neighborhood near a park for less than she would have spent on a similar apartment in University Village. Without Transcendence, Jane would have had to first find the names of neighborhoods near the university and then issue separate queries for each, which could have required her to visit multiple sites and keep multiple browsing windows open in order to compare results. With Transcendence, she needed to perform little more than a single search on Craigslist.

Most attempts to explore the deep web have taken the form of automatic web crawlers [24, 29, 27]. Separate systems have leveraged the structure of web content to enable users to analyze, manipulate and store web information in new ways [5, 21, 15] and other systems facilitate novel mashups of data from disparate sources [8, 32, 19]. Transcendence combines intelligent automated methods with a user interface integrated into the web browser to facilitate an active role for users in the exploration of deep web resources, enabling them to quickly gather, restructure and visualize data from the deep web.

Transcendence offers the following contributions:

- Transcendence enables web forms to be generalized over many values, resulting in simultaneous and specialized web database searches.
- Transcendence integrates the process of filling out web forms with unsupervised information extraction, leveraging a few example inputs to produce many more.
- Transcendence assists users in selecting information that is important to them and exports collected data to tools that enable users to interpret and analyze it.

EXAMPLE USE CASES

Transcendence has many possible applications. Any time there is data hidden behind a web-form that is not easily accessible, our system can help extract it. The following applications are especially compelling.

Aggregated Data

Many websites include user ratings, prices, or other numerical information about specific products or items; however, few sites also include the ability to give aggregated information about these values. For example, consider a movie review website like *imdb.com*. Is a rating of 7 out of 10 a good rating or a poor rating? How are the ratings of movies distributed? How many movies have a perfect 10 rating? These questions would be difficult for the average user to answer as it would require writing a customized crawler and then visiting a number of pages with the crawler to collect data. This crawler is quickly defined with Transcendence.

We used Transcendence to automatically extract the ratings for 2000 movies from imdb.com and to generate a histogram of these values (Figure 2-a). We began the experiment by manually generalizing the search box on the main page using the following four movie titles: "The Matrix," "Rocky," "Scent of a Woman," and "Star Wars." From these initial seed values, the system was able to further generalize to 7142 additional potential movie titles. We ran the extraction process on the top 2000 of these entries, which took approximately two hours to complete. From the collected data, we were able to see that very few movies receive a rating lower than a 3.8/10.0 or higher than 8.8/10.0 rating on *imdb.com*. Also, the average is skewed toward the high end of the scale.

Similarly, the tradeoff between the price of airline tickets and the total time of a flight is often an important purchasing factor for many people, and yet is difficult to extract from existing web forms. Our system can extract a large number of airline fares including the ticket price, the trip time, and the total travel time. These can then be mapped using a scatter plot to give the user interesting information that they can use to determine what is and isn't a good fare. Transcendence generated a scatter plot using data collected during the first task in our user evaluation (Figure 2-b). Notice how the cheapest ticket price does not have the longest flight time!

Mapping Locations

Many sites offer directories of store locations. Access to the directories often involves inputting an address, city, state or postal zip code. Unfortunately, there is rarely a complete listing of all the store locations. Transcendence can generalize across the fields, crawl the results, and aggregate them. It can then map these locations to give the user an overview of all the store locations. Figure 2-c shows a Google map of Best Buy locations (4 per state) automatically generated by our system. The system does not always produce a complete list but quickly provides an overview. For example, it failed to generalize the phrase "Florida" in this example - Montana and Wyoming have no Best Buy stores.

Directory Information

Online directories often attempt to maintain privacy by not including a complete listing of individuals within the direc-

tory. Instead they offer a web form which allows a person to search for a specific individual. Because Transcendence can generalize over names, it can partially reconstruct the underlying databases by issuing queries for the names it generates. The automated nature of the system demonstrates the power of the system and raises legitimate privacy concerns. We tested this by automatically extracting entries from the University of Washington online directory. We first manually generalized the "name" field of the form by entering the common surnames "Allen," "Smith," and "Johnson," suggested by a potential user of the system. We then generalized these examples to 10,063 more possible surnames. While some names produced no results, the resulting extractions overall produced 51,223 unique names and emails.

RELATED WORK

Related work generally falls in the following categories: (i) user interfaces designed to assist users in customizing and improving their web experience and (ii) automated crawlers operating over the deep web.

Collecting and Manipulating Web Information

Transcendence builds from existing systems that enable users to flexibly manipulate, store and merge data from pages that they have identified. User scripting systems, such as Greasemonkey [28] and Chickenfoot [10], modify web pages according to user specifications. Piggy-Bank users can write scripts to extract information from web pages and store that information in a semantic web database [5] and Solvent facilitates the creation of Piggy-Bank scripts with visual tools [6]. Koala uses loose parsing of natural language instructions to make such systems more usable [23].

Sifter automatically parses structured data from web pages to enable advanced sorting and filtering [21]. Transcendence enhances this behavior with semantic tagging of desired fields and enables pages accessible only by querying web forms to be retrieved. Both systems merge collected results and place them in context into the original web page. The visual user interface provided by Transcendence to refine automatically-selected fields in results pages is similar to that provided by existing systems [6, 15, 31]. The interface provided to select, manipulate and assign multiple values to form elements is a novel extension of this idea.

Other systems provide limited support for finding multiple web pages from which to extract data. Web Summaries automatically explores multiple links selected by the user [15] but does not address form submission, required for exploring the deep web. The cloning feature of Clip, Clone, Connect (CCC) enables users to submit multiple form queries and assign different values to them [18]. CCC requires users to manually specify values for cloned input elements in a separate window. Transcendence automatically generates both relevant input values and enables users to refine these choices in the main browsing window.

Many systems seek to provide convenient interfaces that enable web users to "mashup" existing web data in interesting ways. Marmite allows users to select web data and then

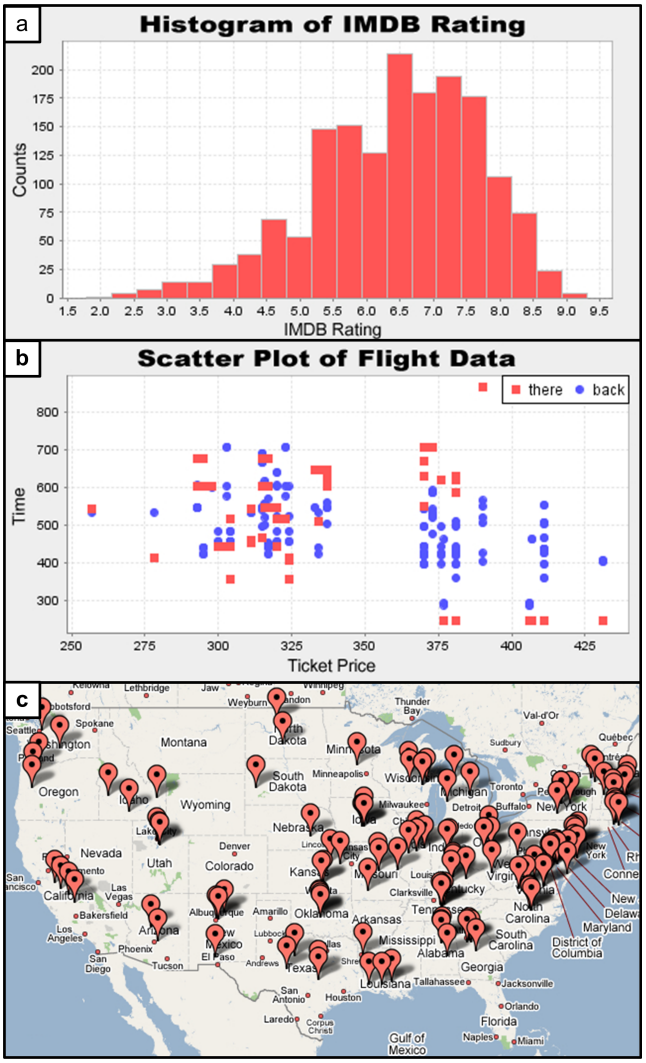

Figure 2. Transcendence Visualizations: a) Histogram of user ratings received by 2000 movies on *imdb.com*, b) Scatter plot showing price on the x-axis and flight time (there and back) on the y-axis, c) Map visualization of Best Buy stores in the U.S.

manipulate it in a familiar interface similar to a spreadsheet [32]. Yahoo! Pipes uses the visual metaphor of a pipe in a program that enables users to connect web resources [8]. d.mix enables web developers to easily copy mashed-up web elements using parametric copies that include not only the data but also the programmatic calls that produced them [19]. Existing mashup systems do not support exploration of the deep web, but they could use the data gathered by Transcendence. Tuchinda *et al.* described how such disparate data sources could be connected using a programming-by-demonstration interface to easily create interesting mashups [30].

Systems that record web macros, such as Creo [17], PLOW [22], and Turquoise [25], could enable users to automatically retrieve pages but cannot generalize user input in order to explore web forms. Creo generalizes text elements included in macros using categorical information from the Open Mind knowledge repository [17] but is not sufficiently general to support many desirable deep web queries. Transcendence

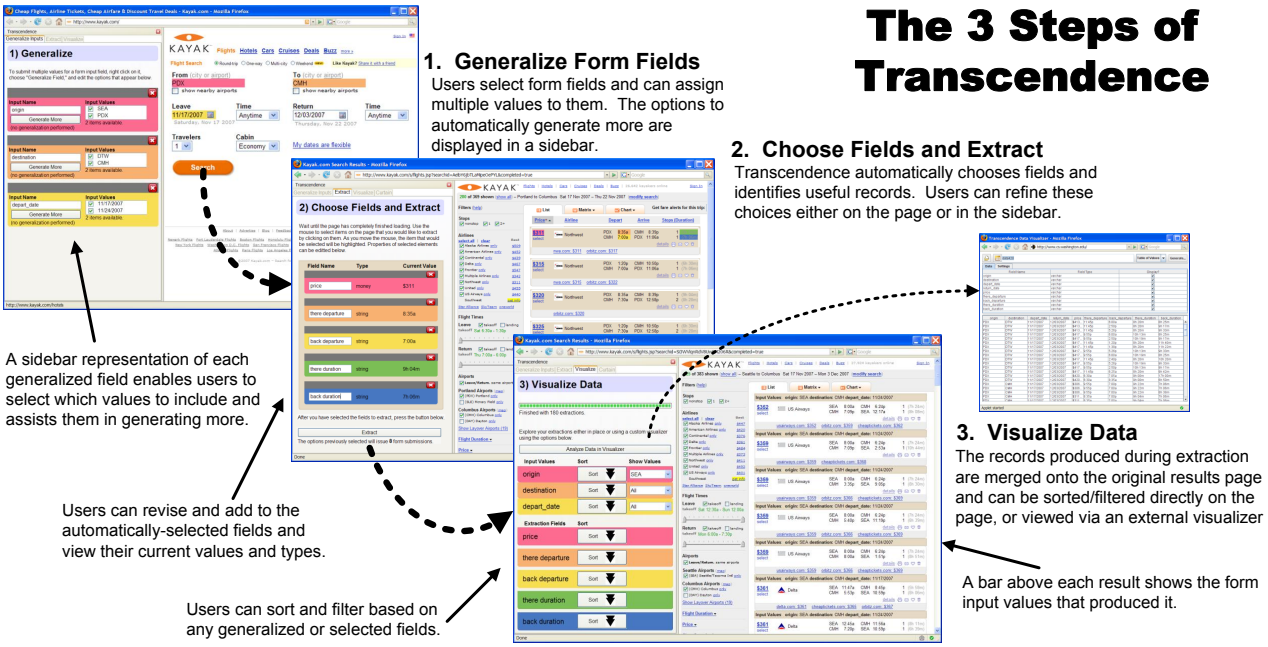

Figure 3. Transcendence consists of 3 steps: Generalize, Extract, and Visualize. Users can generalize forms with multiple values, extract elements of query results for further analysis, and visualize results.

incorporates ideas from web macro systems. Users define forms that they want to explore by searching with that form and can optionally select a link for Transcendence to follow on result pages to bypass disambiguation pages.

Crawling the Deep Web

Existing tools for exploring the deep web resemble traditional surface web crawlers [24]. HiWE [29] and MetaQuerier [13] seek to automatically identify both web forms of interest and the types of their fields. If successful, they submit appropriate values drawn from existing tables in order to produce results pages to crawl. For instance, if a form field is determined to have type "city," then values for it can be found in an existing table of cities. Transcendence does not require appropriate, pre-existing tables and, instead, leverages the implicit types from user input to provide more appropriate lists. For example, a table listing the neighborhoods in Seattle where students attending the University of Washington tend to live is unlikely to exist in a general system, but is incredibly useful for searching on *craigslist.org*.

Ntoulas et al. formalized a method by which both random words and words mined from the results of queries to deep web resources can be used as input to web form fields [27]. This process can be inefficient or impossible if appropriate inputs are not easily found. In Transcendence, users help bootstrap a directed search for relevant values to submit to web forms. Transcendence does not seek to determine the type of the input fields. Instead, it relies on collaborative filtering and unsupervised information extraction to find phrases similar to those input by the user.

This process is difficult to automate because it requires deep web resources to be automatically identified and the schemas representing the structured results that they return to be automatically identified and matched with user queries [14]. To the best of our knowledge, Transcendence is the first system that leverages user input and unsupervised information extraction in order to accomplish the difficult task of finding appropriate input values for web forms.

USER INTERFACE

The Transcendence user interface appears as a sidebar in the browser (a frame occupying its leftmost portion) as well as augmentations to user-chosen web pages (Figure 3). Using Transcendence is logically broken into three tasks: (1) Generalize, (2) Extract, and (3) Visualize. Transcendence first performs each task automatically based on user input, but users can revise these choices, representing a mixed-initiative design [20].

Generalize

Transcendence submits multiple queries for users by generalizing relevant web form fields. To make a text box (or other form field) generalizable, users can right click on the object itself and choose "Generalize Field" (or type CTRL-G) and an area corresponding to the form field appears in the sidebar on the left (Figure 3-1). Participants in a preliminary study of the system had difficulty relating input elements with their corresponding sidebar elements. To assist users, when input fields are focused or the mouse is hovered above them, the corresponding element in the sidebar is highlighted. The user can use the sidebar element to enter several values for the form field and can do this with multiple fields. Users can submit their own values, or they can request that Transcendence find even more similar values automatically (such as more foods or more airport codes) by clicking a "Generate More" button. Once the web form is submitted, the site's

result page is displayed so that the user can define which of the resulting elements will be extracted in the next step.

Extract

In order to allow users to easily extract and collect elements of the results that are important for analysis, Transcendence displays the output of one example query and assists users by automatically selecting relevant elements to be extracted. The automatic selection process occurs in two stages. The first stage uses an algorithm in Sifter [21] to identify the repeating records on a results page. In the second stage, we query the DOM of the first of the repeating records identified in the first stage for text nodes. Text nodes that are contained in links or identified as Transcendence types are automatically selected as fields of interest. Other nodes are selected as fields of interest if they are identified as logical containers of separate sections of text. Transcendence uses the page structure to extrapolate all of the similar elements in the results. For example, if a user decided they were interested in the price and flight duration of one airline ticket on a travel web site's results page, Transcendence will find all of the prices and their respective flight durations for each flight in the list automatically. Users can revise and add to elements automatically selected using a visual interface (Figure 3-2). Transcendence now uses all of the generalizations the user defined in the previous step to extract the relevant elements requested in this step.

Visualize

Transcendence allows users to restructure the output of multiple form queries to best suit their search needs. It displays conglomerated results while maintaining the original page structure (Figure 3-3). Users can change the order in which results are displayed by sorting based on the extracted elements or filtering based on fields that were generalized. For a more in-depth analysis, our visualization tool (Figure 3-3) transforms the data into graphs, scatter plots, histograms, or geographical displays.

TRANSCENDENCE SYSTEM

Figure 4 shows the architecture of Transcendence. The core of the system is a Firefox Extension written in XUL and Javascript. The extension is also responsible for coordinating with the PostgreSQL extraction database, the three automatic generalizers, and the data visualizer. As described in the User Interface section above, gathering data with Transcendence consists of three main stages: generalize input, extraction data, and visualize the results. Here, we describe the system implementation in terms of these three steps.

Input Generalization

In Transcendence, input fields can be *generalized* by assigning multiple values to them. Form elements usually only have one value, which restricts the expressiveness of a form. When searching for airline prices, users may want to fly from either Seattle or Portland to either Newark or Philadelphia depending on the price and other details of the flight. Transcendence allows such extended queries to be expressed on

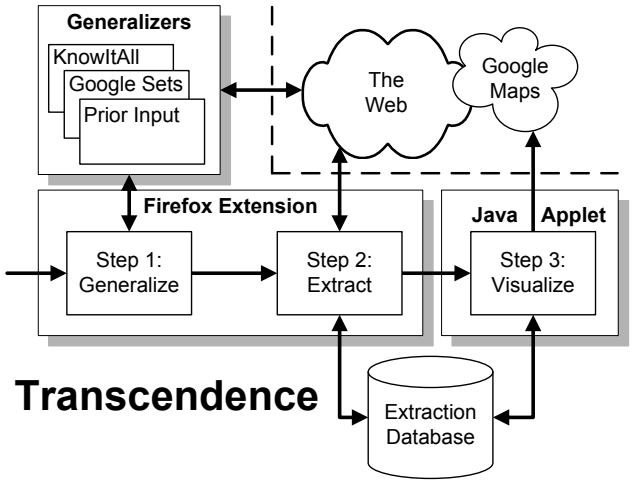

Figure 4. System architecture diagram showing the components associated with each step of using our system

existing web forms. The system allows textboxes to contain multiple strings, checkboxes to be both checked and unchecked, select boxes to be assigned multiple values, and multiple radio buttons in the same group to be checked.

For all elements but textboxes, the space of input generalization is restricted by the element itself. In order to automatically generalize textboxes, Transcendence asks users to provide a few values and uses unsupervised information extraction and suggestions ranked by collaborative filtering to generate a number additional values likely to be appropriate.

Unsupervised Information Extraction

Unsupervised information extraction provides powerful techniques for extracting structured information from unstructured web text. While adept at collecting lists of items, extracting the complex records characteristic of the deep web is still too difficult. Although agnostic to the method used, Transcendence uses KnowItAll [16] and Google Sets [2] for their recall and speed, respectively. Transcendence uses these systems unaltered, but employs them in the novel application of providing inputs for form fields.

Transcendence utilizes the list extraction component of KnowItAll. List extraction proceeds iteratively, first taking as input a series of seed values provided by the user and then using them to generate a number of similar values, which it uses to repeat the process. To extract additional items, the system issues queries to Google for different combinations of the seed values and analyzes the returned pages. If any of these pages contain all of the seed values in a list, then the other items in that list are extracted and saved for future rounds. KnowItAll defines a list as being either comma-delimited or all items appearing in the same column or row of a table. KnowItAll is able to quickly compile large lists of words given only a few instances, provided that these words commonly appear in lists online.

The Transcendence Firefox extension receives a series of seed words entered by the user and sends these to a Know-

ItAll server running on the client. The server first checks its current records; if it has already conducted a generalization run with those seeds, it simply sends the existing values back to the user. If such a run is still processing, the server sends back the generalizations collected so far. If there is no record of such a run, one is started. In this way the results of many searches can be collected and common search results will be readily available.

The Google Sets web service takes up to 5 phrases as input and returns up to 50 similar phrases in return [2]. Although the mechanism powering it is unpublished, it is believed to operate by associating the context of the entered phrases on the web with other phrases found on the web. Google Sets is desirable because it responds quickly, but it is limited to only 50 results. KnowItAll is useful in conjunction for providing large numbers of generalizations. KnowItNow could potentially provide the speed of Google Sets and the recall of KnowItAll, but a public instantiation is unavailable [12].

Previous Inputs as Suggestions

A rich source of phrases are the phrases previously entered by users. Transcendence records the phrases with which users manually generalize form fields and retrieves them later in response to new input. For instance, if a user enters the surnames "Smith," "Johnson" and "Allen" into a textbox, the system can find other surnames such "Anderson," "Harrison," and "Baker" because they appear in similar contexts on the web. If another user later queries the system with "Leung," "Johnson," and "Ordway," then Transcendence can consider returning "Smith" and "Allen" because "Johnson" appeared in both lists.

Phrases retrieved in this way are assigned a weight equal to the number of elements in the intersection of the user-provided list and the lists containing common phrases. Google Suggest provides a similar service for suggesting search keywords [3]. The results are combined with those retrieved using unsupervised information extraction. Such combinations have been explored previously [11].

Data Extraction

The web page that is returned as the result of submitting a web form can either present a single data record, such as the movie information provided by the Internet Movie Database (IMDB), or an array of data records, such as the many flight choices returned by $kayak.com$. Transcendence employs a user-centered data extraction algorithm that can accommodate both cases. When stage two begins, Transcendence automatically selects important fields. Users can then add new fields or remove selected fields from one result set (such as one flight or one apartment listing) until all desired fields are selected. From those selections Transcendence is able to automatically determine the DOM element most likely to represent a complete record and can generalize to detect other records on similar pages.

When a field is selected, the system first determines an XPath for the given node [7]. We assume that the visual layout of the data on the page follows its structural organization, and, therefore, the field is likely to share a common parent in the DOM tree with other fields in the same data record. For pages containing only one data record, that body is the root of the DOM. To extract the likely parent nodes, Transcendence uses a process based on finding the common parent of the fields that maximizes visual layout while preserving the cardinality of the set of matching records [21]. As additional fields are selected (automatically or by users), the system can better estimate which node is most closely associated with each record.

The system next assigns an XPath to each record and a relative XPath from each record to each field contained within it. In order to capture as many records as possible, the system calculates a general XPath to each record that includes only tag names, for example **/html/body/div/table/tr**. To identify each field within a record, the system assigns a more specific XPath, such as **td[3]/b[@class="pricing"]**. When records are matched, at least 80% of the identified fields must also be present. These steps help eliminate spurious matches of fields while remaining general in matching records.

Document Preprocessing

The selection of both fields and records relies on the existence of matching, underlying DOM elements, but a DOM element that uniquely contains only the elements of a single record does not always exist. To improve robustness to this type of failure, Transcendence employs two preprocessing steps that add semantic structure to the existing DOM. First, the system improves pages that express records using sequential patterns of elements instead of using separate DOM elements. Such examples are generally difficult for DOM-based extraction systems. To address this problem we use a recursive process inspired by Mukherjee *et al.* to add additional elements to the DOM to accurately express records characterized by repeating elements [26]. Figure 5 demonstrates how additional <**span**> elements are added as parents of the result of this preprocessing on a simple example in which records consist of an <**h2**> and a <**p**> element. After preprocessing this example, the records can be identified and selected using the new <**span**> elements.

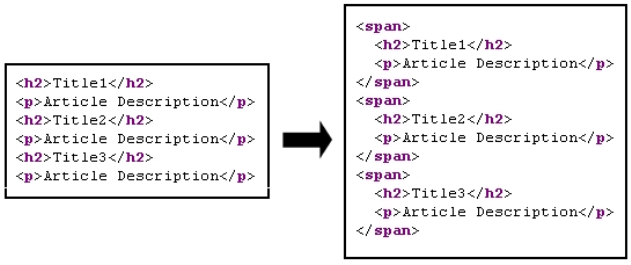

Figure 5. Our preprocessing step inserts additional elements into the DOM model to assist in extracting list items

Promoting text nodes containing common semantic types to element nodes ensures that the fields they represent are selectable. For instance, the text string "User Rating: 7.3/10.0" is contained within a single text node and so the extraction mechanism will allow users to select only the entire string and not just the rating (7.3). Transcendence promotes text

matching certain semantic patterns to selectable elements. Currently, the system supports author, currency, date, number, URL, and acronym types. Other systems enable users to provide either a regular expression that should match text within the selected node [6] or text that will define the region [15]. Sifter automatically chooses an appropriate representation of each field [21]. By exposing low-level data types, Transcendence facilitates easier selection of fields.

Crawling

During the final extraction step, Transcendence submits each combination of the generalized input fields, and extracts any matching records from the resulting pages. The input form is first loaded in a hidden, off-screen browser window and the value of each input field is programmatically set. Before submitting the form, Transcendence calls any **onsubmit** events that are registered with the input form, which often prepares the input data. Transcendence next submits the form and waits for the results page to load. When it does, the page is preprocessed and matching records are extracted.

Transcendence can make extractions from highly-dynamic, AJAX-driven deep web resources. The content on traditional web pages does not change after it is downloaded, but this assumption does not hold for many deep web resources. For example, a search on $kayak.com$ immediately returns an initial results page, but the page updates as additional flights are found. Determining when a dynamic web page is finished updating is intractable, but Transcendence uses simple heuristic that enables it to efficiently extract information from dynamically-updating pages. The system observes the initial search in order to determine if the page content changes and, if so, approximately how long it should expect to wait for it to finish. During this interaction, the system records both the number of page loads that occur and the total time that elapses between when the web form was submitted and the page content stops changing. When the user selects a field, this indicates to the system that the user believes the page has stopped changing. Page changes are recorded by periodically (every 2 seconds) saving a hash value representing the text content of the current DOM and comparing it to the previous recording. Transcendence decides when to stop extracting based on the following three conditions: (i) it has waited 1.5 times longer than the original form submission document took to stop changing, (ii) the maximum number of loads has been reached, or (iii) it has extracted a sufficient number of matching records.

Transcendence uses multiple, simultaneous crawling threads. With ten threads operating in parallel, it can wait up to a minute for each page to finish downloading and still be able to parse a new page of results every six seconds. This offers users a huge advantage compared to waiting for the serial execution of multiple manual queries. For large extractions, the system throttles its extraction rate.

Visualization

Once Data Extraction is complete, the final step is to visualize and utilize the data effectively. Transcendence provides two powerful visualizations methods. First, it can display each recorded extraction in-place in the original web page. The system adds an information bar above each, so users can easily see which input values produced each result (Figure 3). An interface in the sidebar allows users to sort the extracted records based on any of the input form fields or extraction fields. Users may also elect to show only records generated from specific inputs. The results displayed by the system are added to the original results page and are reordered and refreshed based on these options. Future versions of our system will integrate additional in-place visualization inspired by other work in this area [15, 18].

The data gathered by Transcendence can also be incorporated into external programs. The provided *data visualizer* allows users to view the data that they have collected using the following methods: *Table of Values*, *Histogram*, *Bar Chart*, *Line Chart*, *Scatter Plot*, and *Google Maps*. The visualizer is a web-based Java Applet that connects to the extraction database. The user is presented with a table containing a row for each generalized field. Each row has the following columns: field name, field type, and a visualization-specific column. The visualizer uses JFreeChart [4] to build plots and graphs, and the Google Maps API to geocode addresses and display them [1]. The applet connects to the database holding the extracted results, calls the appropriate API to visualize the data, and enables powerful exploration of extracted data.

GENERALIZATION ASSESSMENT

We evaluated the ability of Transcendence to generalize using the KnowItAll list extractor from phrases provided by users because it provides the most generalized phrases in large extractions. Twenty computer science students were asked to provide three examples from several different classes including airport codes, surnames and computer science conferences, which could be useful in searching for flights, directory information and academic publications, respectively. Although the performance of the KnowItAll list extractor has been explored before in the context of automatic information extraction [16], we were interested to see how it performed on a limited number of user-provided examples.

We were most interested in the recall of the generalizing process. Recall is the number of correct phrases produced compared to the total number of correct phrases. While incorrect generalizations will increase the overall time Transcendence requires to run, it will not affect the correctness of the results returned. This is because inappropriate fields will not produce valid results and, in the event that they do, users can easily filter out the incorrect inputs after extraction.

To determine correctness, we used an FAA list of airport codes, the top 10,000 surnames from the U.S. census and a large list of computer science conferences. While not necessarily complete, they are of sufficient size to gauge a reasonable estimate of recall, although reported precision should be, therefore, considered a lower bound. These lists already exist online, but we believe they are representative of other lists users may want to create that are not available online. None of these lists were used by KnowItAll in the generalizing process - they were all accessible only by a web form.

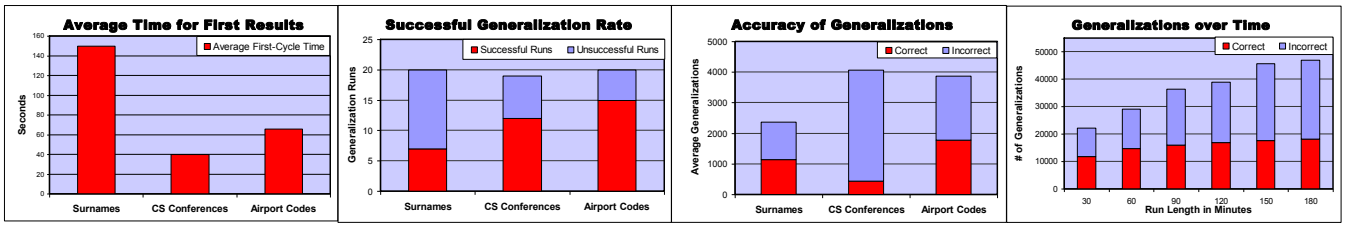

Figure 6. a) Average time to generate the first round of generalizations, or to fail if none are found. b) Percentage of runs that successfully found new generalizations. c) Lower bound accuracy of generalizations per category. d) Lower bound accuracy of a single run of surnames.

We also logged the time required for each run. In the event that the system cannot generalize based on the input seeds provided by the user, the system should fail quickly, so the user can supply additional terms.

The generalization process used the input of participants to collect similar terms and was allowed to run either until it failed to find new extractions, or until a maximum number of three iterative extraction cycles was reached. This restriction was introduced to allow us to perform a large number of searches; in real use, the system could generate many more results. Some runs immediately failed to find similar terms, and so terminated early in the generalization process, typically in less than 30 seconds. Although the typical runtime was around 20 minutes, intermediate generalizations were produced approximately every two minutes.

The first generalizations were returned in about one minute (Figure 6-a). At this point, users could evaluate the returned results and optionally decide to provide more seed terms. Although generalization initially failed in half of the cases, providing a few additional seed inputs is likely to succeed (Figure 6-b). Both surnames and airport codes were extracted with nearly 50% precision, while CS conferences only with 10% (Figure 6-c). This resulted both from CS conferences with acronyms with multiple meanings (which caused the generalization to diverge) and from under-reported precision due to non-comprehensive lists. The number of new generalizations grew slowly after an initial burst (Figure 6-d).

USER EVALUATION

In order to investigate the usability and perceived value of Transcendence, we conducted a study with nine potential Transcendence users ranging in age from 20 to 46 years (6 female). All of the participants had experience with browsing the web. Five described themselves as at least being novice web programmers and the remaining four said they had no web programming experience. We intentionally chose participants with varied programming experience because we hope to ensure that anyone with web browsing (and not necessarily web programming) experience can understand and use Transcendence.

Participants were first given an explanation of the system's functionality and shown an example using $google.com$. The example involved submitting multiple search queries for a person's name: their first and last name with no middle name, with a middle initial, and with their middle name spelled out. Participants were shown how to sort and filter the results, how to generate a table of resulting values, and shown how they could create line graphs, pie charts, or maps of their data if in numeric or geographic form.

We then asked participants to perform three unstructured tasks using Transcendence. The first was to search for an apartment using $craigslist.org$. Participants were asked to generalize the search field and to enter a few neighborhoods in which they might look for an apartment. After they had done this, participants were asked to use the "Generate More" feature to produce more neighborhoods. They next submitted the original form, optionally revised the selection of fields automatically chosen by the system and then ran their extraction.

For the second task, participants used $kayak.com$ to decide which airport would be best for a hypothetical upcoming trip. We told them they could either leave from Seattle (SEA) or Portland (PDX) and arrive at either Columbus (CHM) or Detroit (DTW). Their hypothetical dates of travel were flexible with two different possible departure dates and a specific returning date. The main priority was to find the best price, so long as the total flight time was reasonable to them.

The third task was to visualize the geographic distribution of REI stores across the United States. Participants were asked to go to the Store Finder at $www.rei.com$ and to create a list of all REI stores by generalizing over all states, then map them based on the store address.

After completing the three tasks, participants answered a short, 20-question survey about their experience. Results of the 7-point Likert scale questions can be found in Figure 7. The remaining questions on the survey asked about (1) aspects of the system were hard to use, hard to understand, or not useful, (2) changes to the system participants would like to see, and (3) other potential tasks for which the participants might use this system. Overall, the study took approximately 30 minutes on average.

Nearly all participants agreed that Transcendence is a useful, powerful tool for finding information that would otherwise be difficult to manually recreate (Figure 7). Most thought that it was not at all or only mildly tedious to use. They agreed that they would use it in the future and doing so would save them time. Perhaps due to the shortness of the study, we received mixed results regarding the difficult of learning Transcendence. Most participants thought it was relatively easy to learn, but a handful found it difficult to

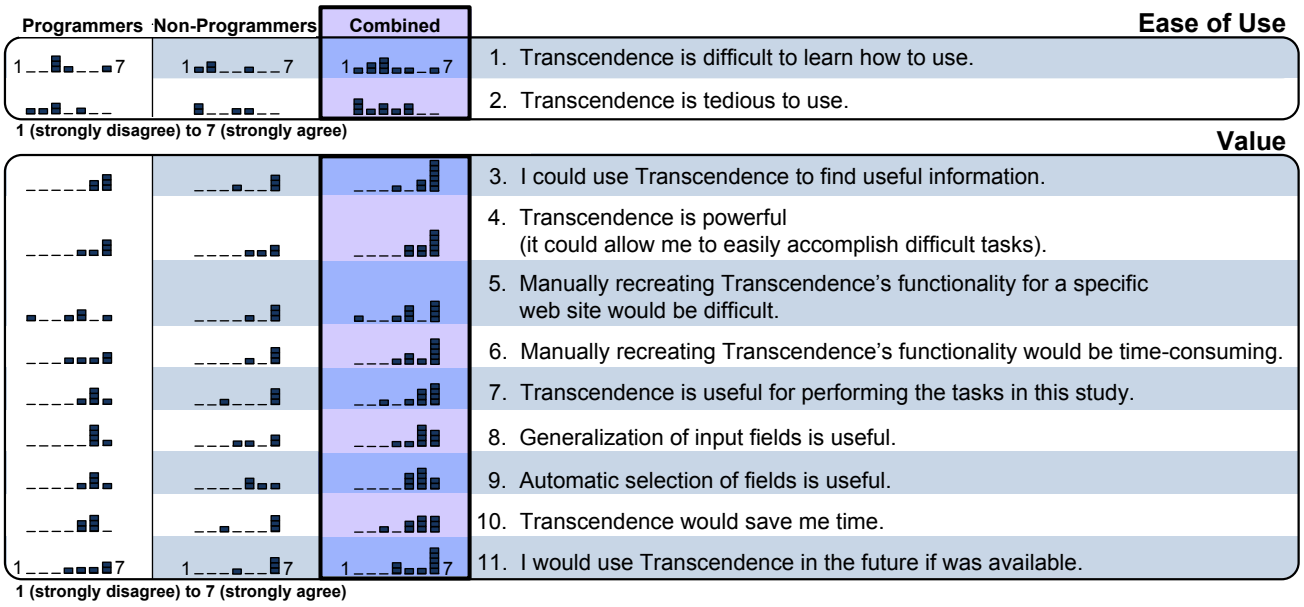

Figure 7. Results of a questionnaire given to participants after exploring Transcendence, indicating that they found Transcendence useful and powerful (3,4,8,9), felt that manually recreating its functionality would be difficult (5,6) and that they would likely use it in the future (10,11).

learn to use the system. Many of these concerns were voiced in the open-ended portion of the questionnaire.

Some participants had difficulty relating the visual elements on the source web page to the query variables in Transcendence. Nearly all participants requested some additional form of transparency: they wanted to see more of what was going on while they waited for queries to be performed and they wanted better feedback regarding failed or unexpected search results. For example, one participant conducted an initial search using states in which no REI stores are located. The search produced no results that could be used to select desirable fields and this was disorienting for the participant. A clear separation, yet convenient integration of Transcendence and source web pages, along with more robust handling of errors, is needed and will be an interesting direction for future versions of the system.

Participants provided many feature requests for future versions of Transcendence. The ability to search multiple web pages (such as both $amazon.com$ and $froogle.com$) was a popular suggestion. One participant suggested the ability to pipeline the results from one search onto the next in a mashup fashion. For example, after using Transcendence to decide on a flight itinerary, one could automatically check the weather forecast in the layovers and destinations.

The generalization feature received mixed reviews. While all participants agreed that it was useful for the tasks in the study, some questioned the value of creating lists not guaranteed to be complete. Several participants were pleasantly surprised during the Craigslist task when a neighborhood in which they would consider living but had forgotten to list was automatically added to the list. In the REI task, the system was able to generalize to all 50 states based on the participants' input of 3-4 states in all but two cases.

Participants liked the automatic selection of fields, which is a nice result given that the time required to do this manually in a previous version of the system had been a consistent complaint. Two participants mentioned that they would have liked to have been able to select the number of bedrooms as a field in the Craigslist task, but this is not currently possible because this information appeared in a block of text not identified as a semantic type.

Many of our study participants could imagine themselves using Transcendence for travel tasks such as searching for and mapping hotels or for comparison shopping. One participant hoped to use it to help search for people who go by both their given name and a nickname, such as both Ben and Benjamin. The participants all found the task that involved searching on $kayak.com$ to be the most compelling and many asked if Transcendence would be made available specifically so they could use it for that task.

CONCLUSIONS & FUTURE WORK

We have created Transcendence and demonstrated its ability to provide a more flexible and powerful interface to the deep web. Participants found the system to be a powerful way to explore deep web resources and have expressed interest in using it in the future.

We plan to further explore the power of using unsupervised information extraction to help users explore the deep web. Many fields on the web are implicitly connected - for example fields for cities and states - and we plan to explore interfaces for expressing these relations. The browser platform may be inconvenient for performing large extractions. After crawlers have been defined by the user, they could be combined to create custom vertical search engines or exported to an extraction platform where extended crawls would be

more convenient. They could also be shared, enabling new users to immediately leverage the crawlers created by existing users.

ACKNOWLEDGMENTS

We thank Mira Dontcheva for invaluable discussions, the University of Washington Turing Center for providing access to the KnowItAll list extractor, and participants in our user evaluation for their creative and valuable feedback.

REFERENCES

1. Google Maps API. http://www.google.com/apis/maps/.
2. Google Sets. http://labs.google.com/sets/.
3. Google Suggest. http://labs.google.com/suggest/.
4. JFreeChart. http://www.jfree.org/jfreechart/.
5. Piggy bank. http://simile.mit.edu/piggy-bank/.
6. Solvent. http://simile.mit.edu/solvent.
7. XML path language (XPath) version 1.0.
8. Yahoo pipes. Yahoo! Inc. (2007). http://pipes.yahoo.com/.
9. Bergman, M. K. The deep web: Surfacing hidden value. *The Journal of Electronic Publishing*, *7*, 1.
10. Bolin, M., Webber, M., Rha, P., Wilson, T., and Miller, R. C. Automation and customization of rendered web pages. In *Proc. of the 18th Symp. on User Interface Software and Technology (UIST '05)*. Seattle, WA, USA, 2005, 163–172.
11. Burke, R. Integrating knowledge-based and collaborative-filtering recommender systems. In *Proc. of the Workshop on AI and Electronic Commerce (AAAI '99)*.
12. Cafarella, M. J., Downey, D., Soderland, S., and Etzioni, O. Knowitnow: fast, scalable information extraction from the web. In *Proc. of the Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '05)*. Association for Computational Linguistics, Morristown, NJ, USA, 2005, 563–570.
13. Chang, K. C.-C. and He, B. Toward large scale integration: Building a metaquerier over databases on the web. In *In Proc. of the 2nd Conf. on Innovative Data Systems Research*. 2005.
14. Doan, A., Domingos, P., and Halevy, A. Y. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proc. of the 2001 ACM SIGMOD Intl. Conf. on Management of data (SIGMOD '01)*. 2001, 509–520.
15. Dontcheva, M., Drucker, S. M., Wade, G., Salesin, D., and Cohen, M. F. Summarizing personal web browsing sessions. In *Proc. of the 19th annual ACM symposium on User interface software and technology (UIST '06)*. New York, NY, USA, 2006, 115–124.
16. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. Methods for domain-independent information extraction from the web: an experimental comparison. In *Proc. of the 19th Natl. Conf. on Artificial Intelligence (AAAI '04)*. 2004.
17. Faaborg, A. and Lieberman, H. A goal-oriented web browser. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '06)*. Montreal, Quebec, Canada, 2006, 751–760.
18. Fujima, J., Lunzer, A., Hornbaek, K., and Tanaka, Y. Clip, connect, clone: combining application elements to build custom interfaces for information access. In *Proc. of the 17th Symp. on User Interface Software and Technology (UIST '04)*. 2004, 175–184.
19. Hartmann, B., Wu, L., Collins, K., and Klemmer, S. Programming by a sample: Rapidly prototyping web applications with d.mix. In *In Proceeding of the 20th Symp. on User Interface Software and Technology (UIST '07)*. Newport, RI, USA, 2007.
20. Horvitz, E. Principles of mixed-initiative user interfaces. In *Proc. of the SIGCHI Conf. on Human factors in computing systems (CHI '99)*. 1999, 159–166.
21. Huynh, D. F., Miller, R. C., and Karger, D. R. Enabling web browsers to augment web sites' filtering and sorting functionalities. In *Proc. of the 19th Symp. on User Interface Software and Technology (UIST '06)*. ACM Press, New York, NY, USA, 2006, 125–134.
22. Jung, H., Allen, J., Chambers, N., Galescu, L., Swift, M., and Taysom, W. One-shot procedure learning from instruction and observation. In *Proc. of the Intl. FLAIRS Conf.: Special Track on Natural Language and Knowledge Representation*.
23. Little, G., Lau, T. A., Cypher, A., Lin, J., Haber, E. M., and Kandogan, E. Koala: capture, share, automate, personalize business processes on the web. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '07)*. 2007, 943–946.
24. Madhavan, J., Halevy, A., Cohen, S., Dong, X., Jeffrey, S. R., Ko, D., and Yu, C. Structured data meets the web: A few observations. *IEEE Computer Society: Bulletin of the Technical Committee on Data Engineering*, *31*, 4 (2006), 10–18.
25. Miller, R. C. and Myers, B. Creating dynamic world wide web pages by demonstration (1997).
26. Mukherjee, S., Yang, G., Tan, W., and Ramakrishnan, I. Automatic discovery of semantic structures in html documents. In *Proc. of the Intl. Conf. on Document Analysis and Recognition (ICDAR '03)*. 2003.
27. Ntoulas, A., Zerfos, P., and Cho, J. Downloading textual hidden web content through keyword queries. In *Proc. of the 5th ACM/IEEE-CS joint Conf. on Digital libraries*. 1995, 100–109.
28. Pilgrim, M., ed. *Greasemonkey Hacks: Tips & Tools for Remixing the Web with Firefox*. O'Reilly Media, 2005.
29. Raghavan, S. and Garcia-Molina, H. Crawling the hidden web. In *Proc. of the 27th Intl. Conf. on Very Large Databases (VLDB '01)*. 2001.
30. Tuchinda, R., Szekely, P., and Knoblock, C. A. Building data integration queries by demonstration. In *Proc. of the 12th Intl. Conf. on Intelligent User Interfaces (IUI '07)*. ACM Press, New York, NY, USA, 2007, 170–179.
31. Turner, S. R. Playtpus firefox extension (2006). http://platypus.mozdev.org/.
32. Wong, J. and Hong, J. I. Making mashups with marmite: towards end-user programming for the web. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '07)*. ACM Press, 2007, 1435–1444.