Linking Messages and Form Requests

Anthony Tomasic
Institute for Software Research International
Carnegie Mellon University
tomasic+@cs.cmu.edu

John Zimmerman
Human-Computer Interaction Institute
Carnegie Mellon University
johnz@cs.cmu.edu

Isaac Simmons
Institute for Software Research International
Carnegie Mellon University
ids@cs.cmu.edu

ABSTRACT

Large organizations with sophisticated infrastructures have large form-based systems that manage the interaction between the user community and the infrastructure. In many cases, when a user needs to complete a form to accomplish a task, the user e-mails a description of the task to the appropriate form expert. In many cases this description is incomplete and the expert engages in a clarification dialog to determine the details of the task. Since many tasks and descriptions are routine, this e-mail dialog can be replaced with an intelligent user interface. The interface proactively reads e-mail (or IM) messages and assists the user in completing the associated task without involving the expert. To ground our vision in a specific application, we have built an agent that functions as a webmaster assistant. For example, a user emails the request: "Change John Doe's home phone number to 800-555-1212" to the agent. The webmaster agent then replies with the biographical data form displaying information about John Doe with the new phone number pre-filled in the form. The user then simply approves the change.

In this paper we describe a prototype website maintenance agent that (i) allows users to express the updates they want to make in human terms (free text input expression of intent), and (ii) allows users to quickly repair any inference errors the agent makes. In addition, we present the results of a proof of concept study that details how interacting with a webmaster agent that makes inference errors is both more efficient (faster) and more effective (errors made to site) than sending a request to a human webmaster. We conclude the paper with a discussion of the application of our work to any form-based system.

Categories and Subject Descriptors: H.5.2 User Interfaces: Interaction Styles

General Terms: Algorithms, Design, Human Factors

Keywords: natural language analysis, machine learning, forms, virtual information officer, VIO, microforms, information extraction

IUI'06, January 29–February 1, 2006, Sydney, Australia.

1. INTRODUCTION

Today, large organizations devote resources to allow their employees to update information in company databases. These organizations provide internal web sites with FAQs, printed documentation, and administrative support help desks accessible via email or phone in order to allow their employees to keep information up-to-date. The following two scenarios provide examples.

Scenario 1: Consider an employee who wishes to modify the automatic deposits to her pension plan. Because she makes this modification rarely, she most likely does not know the appropriate procedure or where to find the appropriate form. In a large organization she can take one of several actions: (i) search the intranet web site for a FAQ that describes the procedure and provides the URL of the form; (ii) call the Human Resources department for help; (iii) delegate this task to an assistant, if she has one; or (iv) ask another employee if they know. Regardless of the approach she chooses, considerable *hunt time* (time to find the correct form) will be spent by her or other employees. To reduce employee effort on this kind of task, organizations devote large amounts of resources to preparing print and web documentation and to maintaining support staff.

Scenario 2: Consider a manager that issues a request to a webmaster support staff member: "Add a new employee, John Doe <John.Doe@company.com>, to the website." In the best case, the webmaster understands the request and has all the information needed to complete the task and notifies the manager of the result. (In this case, the support person's interaction reduces to Scenario 1 above). More often, confusion, ambiguity, missing information, and mistakes abound that result in an "email ping-pong" involving clarification requests. For example, if every person on the website must be associated with a project, the webmaster will reply to the manager's request with a clarification request for additional information.

Our Virtual Information Officer (VIO) is an agent, implemented using two machine-learning (ML) algorithms (described in detail in Section 3) that processes update requests. With the system, users email their update requests to the VIO. The VIO then selects a form, fills it out, and returns it to the user for approval.

This approach offers four advantages: (i) users can state their update intent in free text instead of navigating to a form named by a developer; (ii) replying to an update request with a form reduces ambiguity; (iii) replying with a form makes identification and repair of agent errors easier, and (iv) the VIO agent is available 24-hours a day, seven days a week to make immediate updates.

The VIO assists users in making "point" modifications (expressed in a single form) to websites. Since form developers implement complex transactions as application code associated with a form,

users are effectively executing complex transactions on the underlying database when they approve a form that has been generated through simple free text input.

The prototype demonstrates a novel form of *natural language analysis* by applying *information extraction* technology [7, 8, 16]. Every natural language system has as its core a lexicon, a grammar, and semantic rules. Roughly speaking, the implementation lexicon is composed of dictionaries, a log of past user input, and database strings. The grammar is composed of a set of text extraction models (one model per field per form). The semantic rules consist of a classifier and the business logic that implements the forms.

The prototype presents a completed form in response to a request. The user can then correct the form and thus correct any analysis errors that the agent made. While form-based correction of natural language analysis is not new [17], the prototype implements a novel feedback loop that uses the interaction to improve the machine learning algorithms that form the basis of the analysis. Thus, the prototype learns over repeated use many variations in expression.

The prototype implements *micro-forms*. Micro-forms are forms specifically designed to satisfy particular information intent. For example, in a standard form-based system, a form that updates personal information contains form fields for changing the business address, business phone, home phone, cell phone, home address, etc. Given the information intent "Change John Doe's home phone number to 800 555 1212", the micro-form is a reduced version of the standard form containing the field to change the home phone number, but not other form fields. (Of course, the user must be given a way to access the standard form.)

The novel aspects of the prototype design are driven by two underlying goals. First, the performance of natural language analysis is improved by passive (or minimally invasive) observation of the user experience. This implies that our machine learning algorithms are on-line and use "live" data instead of using a carefully constructed training corpus. Second, the prototype is designed to be domain independent. Currently the only domain dependant part of the prototype is the site design and the database schema. All other aspects are automatically handled – in particular there is no domain specific grammar in the prototype.

The evaluation of the prototype, conducted using behavioral experimental methodology, demonstrates the improved efficiency and effectiveness of a human-VIO method of update as compared to a human-human (webmaster) update. Our research has broad implications since the results apply to *any form system*.

The remainder of the paper is organized as follows. Section 2 describes the prototype and human-VIO interaction. Section 3 details the method of understanding intent expressed in the user request. Section 4 describes the machine learning performance. Section 5 details the experimental design. Section 6 shares the results of the behavioral study. Section 7 offers a discussion of our findings. Section 8 provides an overview of related research. Finally, Section 9 contains conclusions about this proof of concept prototype and a discussion.

1. PROTOTYPE

Figure 1 details the functional flow of information and control in the prototype. The user initiates the process by sending an email request (Figure 2), which gets routed to the analysis module of the VIO. The component replies with a pre-filled micro-form (Figure 3). The user reviews the form and adds any additional information or repairs inference errors made by the VIO. If the agent selects the wrong form, the user can override by selecting the correct form from a pull-down menu, which returns a new form pre-filled with extracted data. When the form is complete, the user can either preview the changes to the web site or she can approve the change, causing the execution module to update the database. The results are then forwarded to the learning module that analyzes the entire interaction and improves the analysis module.

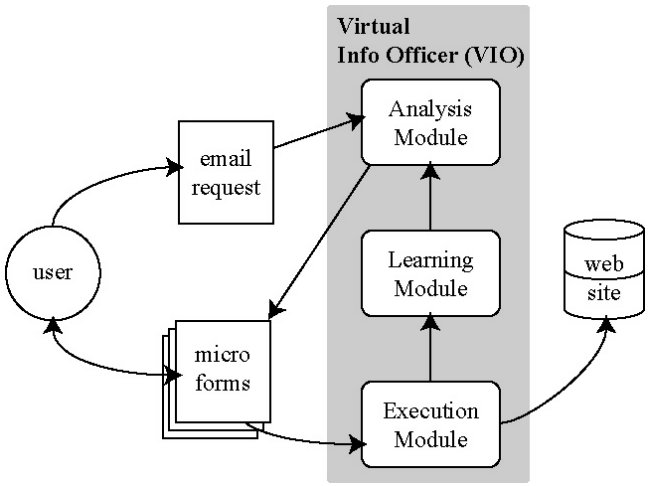

Figure 1: Functional Architecture

The following is a walkthrough example of a transaction with the system. The transaction begins when the user sends the VIO an email request (Figure 2).

The VIO's analysis module matches the request to an "Add employee request" and extracts data for the employee name, email, and telephone. It then replies with an email containing a link to a web form (Figure 3). The user selects the link, launching a browser window that shows the best-guess micro-form completed with the results of analysis (Figure 3). Note that the analysis module has correctly chosen the micro-form and correctly completed 2 of 5 fields that could be completed from the request. The last field, the project of the staff member, cannot be determined from the input. The prototype simply selects a default value in this case. If the VIO had selected the wrong form, the user could override this selection by using the pull-down menu labeled "Change Form…". In addition, before approving this change, the user can follow links at the bottom of the web form to view the web page (or pages) that will be modified and to preview what this page will look like once the update has been made.

We refer to this web form as a *micro-form*, because it only displays the fields related to the user's request. Instead of showing

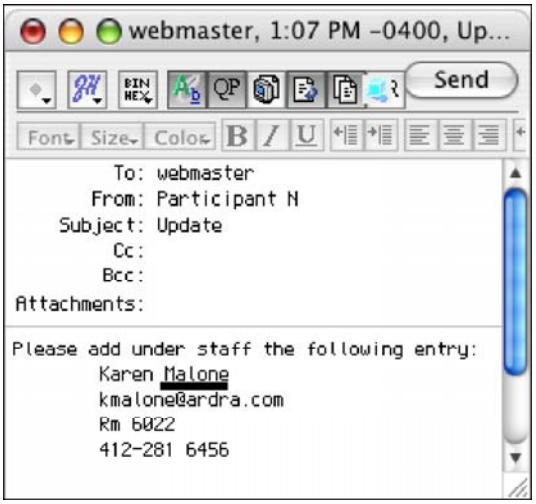

Figure 2: Example Email Request

an entire biographical information form found in most HR databases, the micro-form only reveals the minimum of information for this transaction and thus reduces the amount of navigation time to verify an update. Micro-forms do have a disadvantage. Unlike traditional forms, micro-forms do not elicit additional information not specified in the user's intent (request). We handled this problem by using micro-forms for all tasks except the addition of an entire new entity instance – adding a staff member or adding an event.

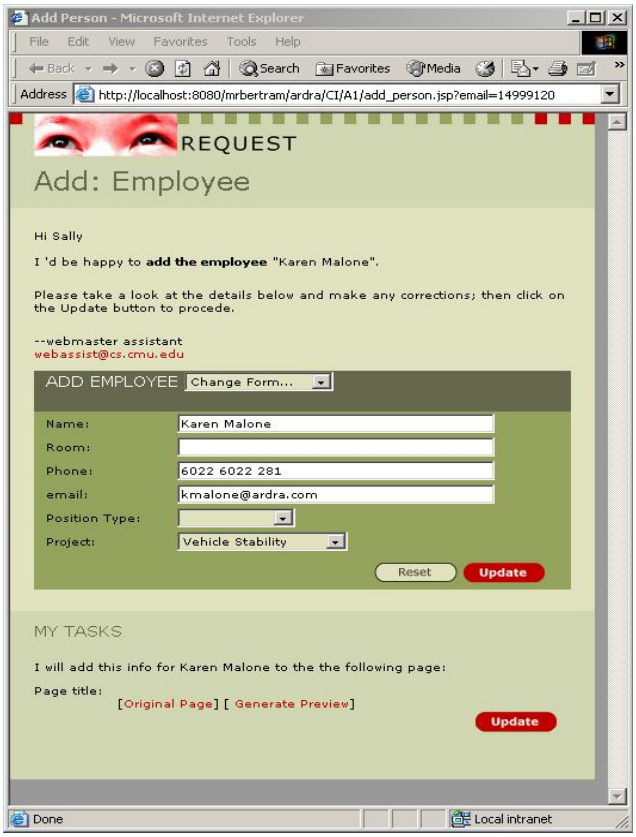

Figure 3: Pre-Filled Form

2. ML METHOD

In this section we describe the details of the method used to analyze requests and select a best-guess form. First, we analyzed, for the set of *k* forms understood by the system, the set of fields that appear in the form. We then developed an *extractor* for each field based on a sequentialized version of Collins Perception Learner algorithm [2,16]. This off-line learning process was trained on data gathered and hand-labeled from pilot experiments described in the Section 5. The result of the algorithm is a learned model for each field in each form. To select the correct form, we trained a k-way boosted decision tree classifier. This classifier chooses between the *k* forms given a modified version of the input.

2.1 Extractor Features

For all extractions, tokens are any sequence of digits, any sequence of alphabetic characters or any single punctuation symbol. Depending on the field, we used a different set of features. For example, for learning names, we used four dictionaries (first names and last names from a white-pages dictionary containing a general list of names and first names and last names from names already in the database), the character type pattern of every current token, plus the patterns of the two tokens to the left and the two tokens to the right of the current token. The character type pattern consists of collapsed sequences of upper and lower case. For example, "jane" collapses to "x+", "John" collapses to "X+x+", "McDonald" collapses to "X+x+X+x+" and "412" collapses to "9+". For learning phone numbers, we used the character pattern for the current token plus the character pattern for 4 tokens to the left and right. The character pattern includes length, so "jane" collapses to "xxxx", "John" collapses to "Xxxx", "McDonald" collapses to "XxXxxxxx" and "412" collapses to "999".

Once a model is trained, it is deployed into the prototype. When a new request input is accepted, it is passed to each model. The model accepts as input the string and returns as output the list of substrings recognized by the model. The first extracted substring is interpreted as the "value" of the field. Thus, for the input string "Delete Jane Doe, listed next to John Doe", the model would extract "Jane Doe" and "John Doe", and "Jane Doe" would be the extracted name in the case. This heuristic works well in this case, but not others, since "After the listing of John Doe, delete Jane Doe" would chose the wrong name. Further work on context is needed here, perhaps with the help of a parser.

2.2 Form Selection Model

The base form classification algorithm uses boosted decision trees. Boosting is a method by which the performance of a *base learner* is improved by calling the base learner again and again on different variants of a dataset, in which examples are assigned different weights in each variant dataset: each new dataset is formed by weighting an example *e* more heavily if *e* was given an incorrect label in previous iterations.

In the prototype uses the "confidence-rated" variant of AdaBoost [4, 10] and a simple decision tree learner that does no pruning, but is limited to binary trees of depth at most five. The decision tree learner uses as a splitting metric the formula suggested by Schapire and Singer as an optimization criteria for weak learners: i.e., split on a predicate *P(x)* which minimizes the function $2\sqrt{W_+W_-}$, where W_+ (respectively W_-) is the fraction of examples

x for which the predicate *P(x)* is true (respectively false). This classifier is converted into a k-way classifier by choosing the class with the highest posterior probability.

The input to the k-way classifier is the set of names of the extractors (i.e, metadata): all extractors that returned at least one value and the classes determined by the classifiers, the names of additional ad hoc classifiers (if they detected something), and additional atoms indicating a successful probe into the database. Hand-coded extractors are used to detect the action (add, delete, replace) of the input and various other references (day of week, month, day, rooms, locations, quoted strings, etc.). This code helps the classifier recognize cases where a user references an attribute without providing an actual value. For example, "Delete the phone number of John Doe" refers to the phone number attribute without providing a phone number.

Another set of analysis uses the output of extractors to generate more features. Each extractor output is used as a lookup into the database (across all attributes). If the value is found in the database for extractor X, the analysis generates an "oldX" feature. Otherwise the analysis generates a "newX" feature. For example, if the *name* extractor extracted a string S, the analysis issues a query against the database to match S with names of people in the database. If the name is found, the analysis generates a "newname" feature, otherwise it generates an "oldname" feature. As a separate step the analysis system resolves references in the input e-mail to database instances. The end effect is an appropriate form that is populated with the correct reference. Note that these "probes" into the database, in the prototype, use exact matching. Soft matching [1] would probably produce better performance.

Table 1: Sample Extractor Training Data

#	Task	Example
1	Add: 2nd phone	Add second phone number: 412 281 4506 for Allen Green
2	Delete: employee	Delete entry for Caldwell McLinn
3	Modify: employee name	Change the name Tim to Timothy
4	Add: employee	Add employee Carl Reese with info reese@ardra.com Rm 6018 412 281 6450
5	Delete: event	Delete the event "Automated Weigh Stations"
6	Modify: sponsor name	Change "Federal Transit Administration" to "Federal Transportation Administration"
7	Delete: 2nd phone	Delete second phone number for Sheu Ng
8	Modify: email	Change "info@roads.ardra.com" to "roads@ardra.com"
9	Add: event	Add an event, "Subcommittee 2.2 Findings" on March 29 at Walters Auditorium

2.3 Training Data

Table 1 displays one example input for each of the nine tasks we tested. Notice that participants almost always use ungrammatical phrases and assume a significant amount of context. The example for Task 3 lists only first names for the task of changing a person's name. The low quality of input is a significant problem for the learning algorithms, however the prototype design compensates for this problem by allowing the user to easily correct any mistakes made in extraction.

3. ML PERFORMANCE

3.1 Extractor Performance

Table 2 lists the performance of name extraction using 5 fold cross validation. The Features column lists two cases, learning with patterns only ("Patterns") verses learning with patterns and the four dictionaries ("+Dict"). Token precision measures the ratio of tokens correctly labeled as part of a name to all tokens labeled as part of a name. Token recall measures the ratio of tokens correctly labeled as part of a name to all name tokens. Span precision and recall are analogous to token precision and recall. F1 is defined as $2pr/(p+r)$ where p is precision and r is recall. The table results show that the learner does a fairly good job of learning names with its dictionaries. The results also show that the learner is biased towards precision vs. recall. Related work [19] describes some additional analysis.

Table 2: Name Extraction Performance

Features	Token Precision	Token Recall	Token F1	Span Precision	Span Recall	Span F1
Patterns	0.72	0.23	0.34	0.72	0.22	0.33
+Dict	0.93	0.84	0.89	0.90	0.78	0.84

3.2 Form Selection Performance

Form selection performance relies on the feature set given to the classifier. The features indicate that a particular extractor generated a result (e.g., feature "time" indicates that the time extractor generated a result). The features "insert", "remove" and "replace" mean that an associated keyword was detected. The feature "keyname" means that the name extractor extracted a name that was found in the database.

The percentage error rate of the form classifier to select the correct form was 20.8% during the actual experiment. Thus, almost 80% of the time the participant is presented with the correct form for her request. While this rate may seem low by machine learning classification standards, it is very high by information retrieval standards (how many times does Google list the page you want in the number 1 spot?). In addition, given that participants in the VIO group outperformed the webmaster group, this result indicates that participants could quickly and successfully use the micro-form to make repairs to agent form selection errors.

4. EXPERIMENTAL DESIGN

This section describes the experimental framework used to measure the human performance tasks using the prototype. The experiment compares a user interacting with the VIO (condition VIO) against a user interacting with a human webmaster (condition webmaster). The experiment measures the time spent to complete the tasks and the accuracy of changes made to the web

site. In addition we wanted to see (i) if the webmaster condition resulted in ambiguous requests that produced email "ping-ponging", (ii) how accurate the agent was at selecting forms, and (iii) how successful users were at correcting agent form selection errors with the micro-form.

Twenty (20) undergraduate student participants were randomly assigned to play one of two roles: requestor or webmaster. For condition VIO, participants emailed nine website update requests to the VIO and then confirmed and approved the updates using the micro-form. For the webmaster condition, participants playing the requestor emailed nine website update requests to a webmaster participant who made updates to a static page website using Macromedia's Dreamweaver (a WYSIWYG web page editing software package). After making the change, the webmaster emailed the requestor a link to a preview of the change. Then the requestor sent a confirmation back to the webmaster to make the change. All participants taking on the role of the webmaster had extensive experience with either Dreamweaver, another WYSIWYG web page editor, or with directly editing HTML.

One of the challenges in this experiment came from instructing the participants about the tasks they needed to complete. The experiment represents nine tasks as a series of web page print outs with hand written edits on the page (Figure 4). This method of task presentation is a common way to communicate change requests [6, 11]. In addition, this method reduces suggestions to the participant for phrases to express the request. Note that the mark up carefully does not use any word (such as "insert" or "add") to indicate the action required. The use of such a word would bias the input phrases generated by a participant. Any such bias has a critical impact on the performance of the extractors and classifiers. However, even this pictorial method introduces some bias. Figure 4 uses "rm" instead of the word "room". In almost every case, participants used this abbreviated form in place of the word "room."

Figure 4: Example Task

After completing a warm-up task, each participant in the requestor role completed each of the nine tasks listed below in Table 3. This table also lists the field information the VIO is able to extract. Participants often provided information that was not used by the VIO. For example, participants often mentioned the name of the web page for the modification. However, the VIO never uses page name in its analysis.

Table 3. Task and Fields Extracted

#	Task	Fields
1	Add: 2nd phone	name, phone
2	Delete: employee	name
3	Modify: employee name	old name, new name
4	Add: employee	name, email, room number, phone number
5	Delete: event	month, day, title
6	Modify: sponsor name	old sponsor, new sponsor
7	Delete: 2nd phone	name
8	Modify: email	old email, new email
9	Add: event	month, day, title, location

Requestor participants in both groups issued an email for each task. Because of the time needed for both the VIO and the webmaster to complete the task and then send a reply, requestors typically sent several email requests before viewing the first reply and completing a single task. In order to compare the performance of the two groups, we tracked the total time to complete all of the tasks, errors made by the VIO, and the errors made by participants in updates to the final web site. In addition, software recorded the screen activity of every participant.

During pilot testing of the experiment we discovered that paying a flat rate to participants lead to an unacceptably high rate of errors made by participants in the VIO group. (No such error was observed in the webmaster condition.) Therefore, we used a bonus for completion of all tasks without any errors as an incentive to check the details of each task transaction.

5. EXPERIMENTAL RESULTS

Table 4 summarizes the results of the experiment. The table lists the experimental condition (**Condition**), the number of requestors (**Re**), the percentage of errors that appear on the website due to participant's activity (**Site Error**), the ratio of emails sent to the optimal number of emails (**Email Ratio**—only relevant for the webmaster condition), and the average of the times taken for each participant to complete the tasks (**Task Time**). Under Task Time we also report the ratio of the task time to the webmaster group task time. This ratio computes the performance increase of the VIO condition compared to the webmaster condition.

Table 4. Time and Error Performance Comparison

Condition	Re	Site Error	Email Ratio	Task Time
Webmaster Condition	12	5.5%	1.22	33.0 min (1.0 x)
VIO Condition	8	2.9%	n/a	22.6 min (1.46 x)

Note that the number of participants is too small to make reliable statistical statements. However, the table does reveal several trends we believe reflect true differences as opposed to illusion from random effects.

The Task Time column indicates that the human-VIO condition outperforms the human-webmaster condition with respect to average time. This result occurs for two reasons: (i) ambiguity in the human-agent communication is reduced through the use of the form that clearly communicates the required information for an update without a back and forth email exchange; and (ii) an agent can select the form and fill it in with extracted data faster than the webmaster can make the point change to the website using a tool. Also, the webmaster condition requires two people to accomplish a task compared to a single individual on the VIO group, so the benefits of reduced task time should be doubled when converting this benefit to a cost reduction calculation. In this case, the VIO shows a 66% reduction in employee-time (32% reduction for requestor and 100% reduction for webmaster).

More surprising is the fact that the site errors for the webmaster group condition were slightly higher than the VIO group. We expected the opposite result for two reasons. First, in the webmaster condition, every task had two people examining the changes, thus raising the likelihood that errors are caught. (This diligence from both participants in checking the details of each change may also have contributed to the longer time taken for this condition.) Second, a completed correct form is difficult to distinguish from a completed incorrect form in some cases. In subsequent work, we have found no difference in the error rate of users between agent-based and non-agent. The difference here may be related to a slightly different incentive structure in the two conditions.

The email ratio column of Table 4 provides some insight into the problem of communication between participants in the webmaster condition. Participants generated many emails above the optimal in this condition. The optimal number of emails is 3 per task: task request from requestor, webmaster reply with preview link, and confirmation from the requestor.

To further understand this communication overhead, Table 5 lists the results of an analysis of the messages in the webmaster condition. The table lists the participant (**Part**), the task number (**Task**), the number of emails (**Emails**), and a classification of the interaction for all non-optimal interactions between participants (**Classification**). The classification codes are as follows:

Incomplete Request: Requestor did not send enough information for the webmaster to complete the task in the initial email. (5 of 108 transactions)

Request Incorrect: Requestor asked the webmaster to complete the wrong task. Correction made after confirmation sent to requestor. (1 of 108 transactions)

Acknowledge: Webmaster emailed acknowledgement after receiving confirmation that the task had been completed correctly from the requestor. (4 of 108 transactions)

Webmaster Incorrect: Webmaster made an error completing the task and it was caught by the requestor and fixed by the webmaster. (1 of 108 transactions)

Inter-task: Webmaster and requestor confused two tasks and resolved this confusion through multiple email exchanges. This is due to the fact that the requestor sends several requests before getting the first reply from the webmaster. (1 of 108 transactions)

Ambiguity: Initial request is ambiguous (not just incomplete) and requires a clarification dialog. (2 of 108 transactions)

Table 5. Email Iterations (14 of 108 transactions)

Part	Task #	Emails	Classification
P1.1	Add: 2nd phone	8	Incomplete Request
P1.2	Add: 2nd phone	7	Incomplete Request
P2.2	Add: 2nd phone	5	Incomplete Request
P1.5	Add: 2nd phone	5	Incomplete Request
P1.5	Add: event	5	Ambiguity
P1.2	Delete: employee	5	Webmaster Incorrect
P2.1	Delete: employee	4	Acknowledge
P1.1	Delete: 2nd phone	6	Request Incorrect
P2.1	Delete: 2nd phone	5	Acknowledge
P2.1	Delete: event	5	Acknowledge
P2.2	Delete: event	5	Incomplete Request
P1.2	Delete: event	4	Acknowledge
P1.5	Modify: email	10	Ambiguity
P3.2	Modify: sponsor name	11	Inter-Task

6. DISCUSSION

The prototype implements a form of the information intent vision defined in the introduction. Clearly for the task at hand, using the agent is more efficient and more effective than using another human to accomplish the same tasks, even when the agent makes errors. While restricted to website updates, our work indicates at least the potential for good results in other web-based applications.

The prototype and experimental framework implement and test only a limited set of all possible updates to a website. Clearly there is an engineering trade-off between the effort put into capturing a particular class of changes as a form and leaving the changes to an engineer. The set of updates the prototype implements are representative of typical website changes.

The performance of the prototype depends in part on the quality of the extraction. We use a sequential learning algorithm that relies on the local context around the target to extract. So extraction works well when the local context provides strong evidence for the extraction: names, phone numbers, room numbers, etc. have a good chance of correct extraction because the local context of a few tokens proves strong evidence. For example, learning a phone number is helped by features such as: the current token is a number, the previous token is a parenthesis, the next token is a parenthesis, etc. Longer strings such as event titles are more difficult to extract if the user does not use quotation marks around the event title. We have not yet attempted to extract paragraph length fields. Very large inputs are generally communicated in the form of email attachments (such as adding a new publication to the website), files in a shared repository, or through a web site URL. In these cases, the file both is classified into a particular type of document (publication, resume, agenda,

minutes, talk abstract, etc.) and extraction is done on the document (extract publication authors, etc.) as an extension to the extraction and classification done on the request itself.

The prototype only deals with a few forms, but large-scale form-based web systems involve hundreds if not thousands of forms. This type of large-scale classification is an open research problem. However, we believe that classifiers may perform well because (a) information intent by users is very precise and (b) forms differ radically in structure and content.

Finally, the prototype is completely handcrafted for the particular application in mind. Each extractor and classifier is labor intensive to construct because pilot training data must be generated and considerable experimental work must be performed to optimize precision and recall. The quality of the generated model depends entirely on the amount of training data, the features used, and the learning algorithm. What is the cost of development of a large number of extractors and classifiers for a new application? Can this cost be significantly reduced? What algorithms best achieve a lower cost? How can these models adjust to changes in the underlying (database) application? How can these models adjust to changes in user behavior?

7. RELATED WORK

Information intent touches on many areas of research. In the natural language processing for database question answering area has a long history [15]. The ASK system [14] provides a natural language interface to updates (at the data and metadata level) and provides a form-based data entry system. However, the system does not appear to fill in forms based on natural language, nor does it provide navigation to a form – the user must know the name of the form of interest. Also, this paper does not discuss errors of any type. Recent work [3, 12] has demonstrated new approaches to this problem that may apply to information intent.

Information intent also touches upon research in Computer Support for Cooperative Work. Grudin [13] discusses difficulties in the successful user adoption of CSCW applications. The paper states that a CSCW system must provide benefit to all users that participate in the system. In applying the prototype to the CSCW case, we have several choices based on the routing of messages. In the current single user prototype, the information intent message, the clarification dialog, and the final approved form are all done by the same user. Instead, in the case of a busy executive, the clarification dialog can be handled by an office assistant and then forwarded to a webmaster. (The office assistant typically has access to the additional information needed to complete a request.) Or, the executive can execute the clarification dialog herself and forward the final approval form to a webmaster. In each of these cases, the benefit and costs of using the system are shifted between users. The best organization is an open question.

The MANGROVE project [5] is targeted at bridging the chasm between the document unstructured world and the database structured world. This project implements a direct manipulation interaction style markup tool for documents. Our prototype may be applicable to the same scenario.

Lockerd, et. al. [9] describe a user study of e-mail based requests to a web master for changes to a web site. We borrowed the before image / after image technique from this paper. The paper reports that detecting delete and update requests exhibited a "semantic pattern" 85% of the time. The data from the reported experiment was used to implement a hand-built parser that understood requests fully 65% of the time.

Interface design where forms are a response to free text input has a history in human-computer interaction, e.g. [17], but these systems do not apply machine learning techniques and thus do not improve with continued use.

Meng [18] reports experimental results on removing the ambiguity between field values and the use of the values in filling out a form. The method involves mapping field references to potential field values and then weighing the reference via a weighted n-gram vector cosine function. This method deserves further investigation, in particular its application to new values that do not appear in the database.

VIO operates with free text as input. In some scenarios, VIO may operate in an environment where the generation of the input document occurs in a controlled way. In this case, user interface techniques (e.g. [20]) may improve analysis of the input.

8. CONCLUSION

We examined the problem of issuing a request to update a website. We used text extraction machine-learning algorithms, classification algorithms, database retrieval and user interfaces to provide a solution. We developed a prototype agent that, using these algorithms, processes commands. We then measured the performance of the agent with users in a controlled experimental environment and showed that the agent helps the user complete tasks faster and with fewer errors when compared to a pair of users accomplishing the same task. These preliminary results indicate that information intent is a real problem, information intent requests can be effectively processed, and that machine learning can help with solutions to this problem.

Information intent requests differ from questions in information retrieval or question answering systems. In these classes of systems, users describe an information need for a question that the user cannot answer. Users expend additional effort to verify the answer. In information intent, users describe a desired result or command. Users can examine the results of the information intent prototype and readily determine if they are correct or not, with little additional effort or information.

Information intent is not limited to the point changes listed above. In response to a broad request, a VIO can provide all of the information, tools, and resources needed to complete standard tasks. For example, a user might tell the VIO that they need to organize a conference, and the VIO can return all of the forms, procedures, and related information needed to complete this task. Thus, information intent is a broad research area.

The prototype uses a novel application of text extraction technology to process information intent commands – essentially applying text extraction to the natural language understanding problem.

The prototype also demonstrates a novel form of user interface interaction design where an information intent request generates a micro-form as an answer to the request. The use of information intent allows the prototype to generate a "micro-form" that focuses precisely on the needs of a particular command. Micro-forms allow more rapid correction of agent errors.

Experimental results show that one person interacting with the VIO completes tasks significantly faster than two people working together. Performance comes from a reduction in ambiguity in the dialog, in the speed with which the VIO analyzes the request, and even though the VIO made form selection errors, the speed with which the user corrects VIO errors. In addition, users could successfully use the micro-form interface to rapidly repair agent form selection errors.

In addition, human errors in updates to databases appear roughly the same for both the person to VIO case and the two people working together case. This area needs more research as we expect some agent errors may be harder to perceive since they may be buried in a pre-filled out form.

8.1 Future Work

We plan to apply the techniques of this paper to other domains to test if the algorithms generalize in a domain independent way. This process will provide some insights into the best architecture and engineering method for the creation of a new information intent system. Integration of other natural language techniques, such as parsing, must also be addressed in this process.

A central functionality for this generalized system is the ability to semi-automatically generate micro-forms. In addition, we would like to formalize the ad-hoc code we now use for augmentation of extracted results. In addition, we use a variety of ad-hoc code for the translation of results into specific presentation style of a form. For example, if the extracted field is associated with a database attribute that (a) is an enumerated type, (b) does not allow updates, (c) the type has a few possibilities, then a pull-down menu is a more appropriate presentation style than a text field. For example, employees may be only full-time or part-time. Selecting this option via a pull-down menu is far better than a text field. Semi-automatic generation of this kind of form presentation style would simplify form construction.

Acknowledgements

Ellen Ayoob, William Cohen, Jason Cornwell, Kyle Cunningham, Susan Fussell, Robert McGuire, Ken Mohnkern and Aaron Spaulding have contributed to various aspects of VIO.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

9. REFERENCES

[1] William W. Cohen, Data Integration using Similarity Joins and a Word-based Information Representation Language, ACM Transactions on Information Systems (18)3:288—321, 2000.

[2] Michael Collins and Yoram Singer, Unsupervised Models for Named Entity Classification, Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP99), College Park, MD, 1999.

[3] Oren Etzioni, Alon Halevy, Henry Levy, and Luke McDowell. Semantic Email: Adding Lightweight Data Manipulation Capabilities to the Email Habitat. International Workshop on the Web and Databases (WebDB), June 12-13, 2003, San Diego, California.

[4] Yoav Freund and Robert E. Schapire Experiments with a New Boosting Algorithm, International Conference on Machine Learning, pp 148-156, 1996.

[5] Alon Halevy, Oren Etzioni, AnHai Doan, Zachary Ives, Jayant Madhavan, Luke McDowell and Igor Tatarinov, Crossing the Structure Chasm. Conference on Innovative Directions in Research (CIDR 2003)

[6] Nathan Halstead. Personal Communication, 2003.

[7] Subbarao Kambhampati and Craig A. Knoblock, editors, Proceedings of the 2003 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico, August, 2003.

[8] John Lafferty, Andrew McCallum and Fernando Pereira, Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, Proceedings of the International Conference on Machine Learning (ICML-2001, Williams, MA, 2001.

[9] Andrea Lockerd, Huy Pham, Taly Sharon, and Ted Selker, Mr. Web: An Automated Interactive Webmaster. CHI 2003.

[10] Robert E. Schapire and Yoram Singer, Improved boosting algorithms using confidence-rated predictions. Machine Learning, 37(3):297-336, 1999.

[11] Aaron Spaulding. Personal Communication, 2003.

[12] Tessa Lau and Daniel Weld, Programming by Demonstration: An Inductive Learning Formulation. Proceedings of the 1999 ACM International Conference on Intelligent User Interfaces, 1999. (IUI'99)

[13] Jonathan Grudin, Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. Proceedings of the 1988 ACM Conference on Computer-Supported Cooperative Work, Portland, Oregon , 1988

[14] Bozena Thompson, and Frederick Thompson, Introducing ASK, A Simple Knowledgeable System. Proceedings of the First Conference on Applied Natural Language Processing, Santa Monica, California, 1983.

[15] Elaine Rich, Natural-Language Interfaces. IEEE Computer Magazine, September, 1984.

[16] MinorThird. http://minorthird.sourceforge.net

[17] Philip Cohen, et. al., Synergistic Use of Direct Manipulation and Natural Language. CHI 1989

[18] Frank Meng, A Natural Language Interface for Information Retrieval from Forms on the World Wide Web. Proceedings of the 20th International Conference on Information Systems, Charlotte, North Carolina, 1999.

[19] William Cohen, Einat Minkov and Anthony Tomasic, Learning to Understand Web Site Update Requests, IJCAI, 2005

[20] Richard Power and Roger Evans. WYSIWYM with wider coverage. ACL-04, 2004.