Learning Information Intent via Observation

Anthony Tomasic
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA
tomasic@cs.cmu.edu

Isaac Simmons
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA
ids@cs.cmu.edu

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

ABSTRACT

Workers in organizations frequently request help from assistants by sending request messages that express information *intent*: an intention to update data in an information system. Human assistants spend a significant amount of time and effort processing these requests. For example, human-resource assistants process requests to update personnel records, and executive assistants process requests to schedule conference rooms or to make travel reservations. To process the intent of a request, an assistant reads the request and then locates, completes, and submits a *form* that corresponds to the expressed intent. Automatically or semi-automatically processing the intent expressed in a request on behalf of an assistant would ease the mundane and repetitive nature of this kind of work.

For a well-understood domain, a straightforward application of natural-language-processing techniques can be used to build an intelligent form interface to semi-automatically process information-intent request messages. However, high performance parsers are based on machine-learning algorithms that require a large corpus of examples that have been labeled by an expert. The generation of a labeled corpus of requests is a major barrier to the construction of a parser. In this paper, we investigate the construction of a natural-language-processing system and an intelligent form system that *observes an assistant processing requests*. The intelligent form system then *generates a labeled training corpus by interpreting the observations.* This paper reports on the measurement of the performance of the machine-learning algorithms based on real data. The combination of observations, machine learning, and interaction design produces an effective intelligent form interface based on natural language processing.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *interaction styles, natural language.*

General Terms

Algorithms, Experimentation, Human Factors

Keywords

Information Intent, Weak Labeling, Domestication

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

1. INTRODUCTION

Workers in organizations frequently request help from assistants in accomplishing tasks. For example, a worker might send to a webmaster a request such as "Please change Leah Davies' phone number to 555-1212." Typically, to process this request, the assistant would perform four steps: (1) find and select the "Change Person" form, (2) instantiate the form with the *target instance* by selecting "Leah Davies." (3) provide the correct information by modifying the phone number on the instantiated form to the new value, and then (4) submit the updated form to the backend website processing system for processing. The backend system executes a transaction and thus the task is accomplished by the assistant on behalf of the user.

In the above example the user's request to an assistant is an expression of the *intent* of the user. With information intent, users express the intent and any additional information necessary to complete a task. This results in a transaction (a change in state of the underlying system) or a workflow execution (a sequence of queries and transactions). This model implies that the domain of discourse is well understood by both parties. By contrast, users expressing an information need to an information retrieval or question-answering system do not need to understand the domain well. Satisfying this need does not change the state of the system, and an information need typically covers a poorly defined or even arbitrary domain.

To understand the information intent of the user in a well-understood domain, a straightforward application of natural-language-processing techniques can be used. However, high-performance parsers are based on (1) domain-specific engineering and (2) on machine-learning algorithms that require a training corpus—a large collection of requests that have been consistently labeled by one or more experts. The generation of a labeled corpus of requests is expensive in time and in effort because requests must be gathered and consistently labeled. In addition, the performance of the resulting parser deteriorates over time. Domains of discourse continually change and expand. In response to this problem, more requests must be acquired and labeled to keep the parser up-to-date. Both the domain specific engineering and the generation of a corpus are major barriers to the construction and maintenance of a parser.

In this paper we propose and evaluate a solution to the problem of constructing an up-to-date corpus. The solution is based on the fact that, during the normal processing of information intent, the assistant generates a long sequence of observable *actions*. We term these observations "wild labels" since they are acquired "in the wild." Wild labels provide evidence for the generation of the *weak labels* of a corpus by the system. Weak labels are distinguished from *gold labels* generated by a human expert. The generation of weak labels from wild labels is called

domestication. The weak-labeled corpus is used to train a collection of machine-learning algorithms. The algorithms generate models that predict the sequence and parameter values of the actions an assistant will take on a new request. The predictions are used to create an *intelligent form system* that aids an assistant in processing information-intent requests.

Our investigation broadly covers the machine-learning aspects and human-computer interaction (HCI) aspects of our proposed solution. To better understand the machine-learning aspects, three issues are studied in this paper.

1. What is the quality of weak labels compared to a gold standard?
2. How well do machine-learning algorithms perform when trained with weak labels?
3. What is the impact of domain-specific engineering?

The first two issues are studied empirically using real-world data. To study the third issue, we restrict our investigation to a system that uses minimal domain-specific engineering. Thus, our reported empirical results are a *lower bound* on performance, since additional domain-specific engineering can be used to improve performance. In addition, we restrict our investigation to assistants that have no training with respect to machine-learning labeling nor do they receive any special training with respect to the system. In fact, assistants of the system have no understanding that machine learning is actually occurring.

There are many HCI aspects to the solution—for example, how are predictions used to generate suggestions to the assistant? How are errors in predictions efficiently detected and repaired by the assistant? Since at system start time, it contains no observations and thus no predictions, how does the interaction evolve as more predictions (and more accurate predications) are generated? While some of these aspects are discussed here, this paper focuses on machine-learning issues. A companion paper [7] describes extensive human-subject experiments and keystroke-level-modeling analysis that demonstrate a significant increase in the assistant's performance. In summary, for exactly the same real-world data and machine-learning algorithms described below, our intelligent form system provides an average 17% reduction in the time required to process information-intent requests, compared to a standard form system.

To empirically validate our solution, we implemented an end-to-end intelligent form-system prototype, the Virtual Information Officer (VIO), which aids assistants in possessing information-intent requests. VIO aids the assistant in solving the first three problems described in the running example: (1) selection of the form for the request, (2) selection of the target instance for the form, and (3) completion of the form. VIO does not automatically submit the form, since the underlying machine-learning models sometimes incorrectly predict the correct action to take.

VIO solves the first problem using a classification machine-learning algorithm to rank the set of forms according to the likelihood that the form satisfies the request. VIO solves the second problem using a reference-resolution algorithm. VIO solves the third problem using an information-extraction algorithm. All these algorithms are trained with weak labels generated by the observed interactions between the assistant and VIO.

The remainder of the paper is organized as follows. Section 2 describes the information transferred and analyzed in each step of VIO. Section 3 describes the algorithm that generates a ranked list of candidate forms. Section 4 describes the analysis of requests with respect to the fields offered in set of forms. Section 5 describes the reference-resolution algorithm. Section 6 describes the algorithm that generates training data from observed actions. Section 7 describes our experimental framework for the evaluation of these algorithms. Section 8 describes the results of the evaluation and discusses the implications of these results. Section 9 discusses related work. Section 10 concludes the paper.

2. Intent Analysis

Figure 1 illustrates the functional steps VIO takes in the analysis of information-intent requests. First, a boosted decision tree model [21] is applied to the request to rank the forms in order of likelihood. In our running example, a correct model would rank the "Change Person" form first. Second, for each field of each form, a conditional random-field [20] model is applied to the request to extract possible new field values. In our running example, a model for a first-name field would extract "Leah," "Davies" for a last-name field, and "555-1212" for a phone-number field. Third, a reference-resolution model generates a ranked list of target entity instances. (Reference resolution is based on a variation of Fellegi-Sunter [25].) In the example, this list would be the instance identifier of Leah Davies' personnel record.

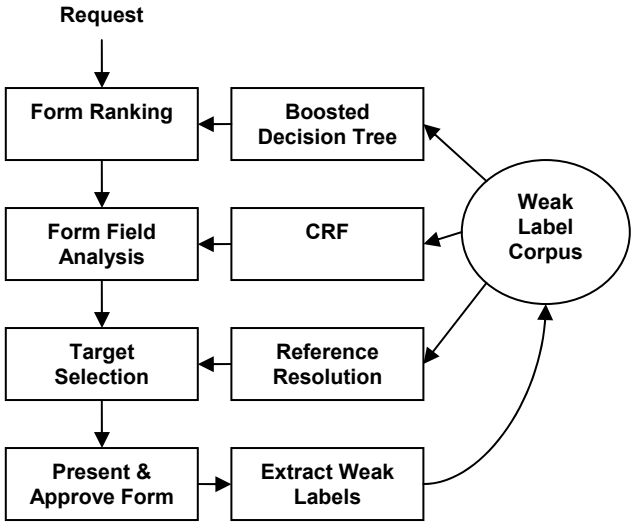

Figure 1: Functional Architecture of VIO

Once the analysis is complete, VIO uses the predictions of the models to aid the assistant through the form system. First, VIO augments the incoming email request from the user with a short ranked list of predicted forms, followed by a structured list of all possible forms. The assistant selects a form. VIO then fetches the predicted target-entity instance and populates the selected form with the corresponding information. In the case that the model has low confidence in its prediction, VIO does not fetch an entity instance. VIO then augments the form with suggestions for changes to fields based on the extraction models. That is, VIO

presents a form with pre-filled suggestions for the relevant update fields of the form. In the case that the model has low confidence, no suggestions are made. The assistant then simply inspects the pre-filled form and inserts missing values and corrects errors. The assistant then submits the form for processing. Note that, in the case that all models have low confidence; the assistant's interaction with the form system is essentially identical to a classical form system with no intelligence. As the confidence of the models rise, the assistant gradually shifts from processing a form in the normal way to checking the work of VIO. Eventually, when the models are highly accurate, the assistant only checks the work of VIO and submits the form.

For every request, VIO observes the following actions: the form selected, the targeted entity-instance selected, and modifications to values on the form. The domestication algorithm takes these observations and generates weak labels in the corpus. This weakly labeled corpus is used to train new versions of the models.

The weak-labeling feedback loop eliminates the cost of generating training data. In fact, the assistant is unaware that any labeling is actually occurring. In effect labels are generated "for free" as opposed to the expensive generation of an expert "gold standard" labeled corpus. In addition, weak labels automatically adjust to modifications in the form. If a new form, field, or entity is added, e.g., "cell phone," VIO automatically creates and tracks labels for the new field. Thus, in keeping with our goals, VIO is almost completely domain independent—almost no domain-specific natural language processing is done (the only exceptions are a common-first-name dictionary and a common-last-name dictionary). Thus, VIO generalizes automatically, without domain-specific engineering or labeling, to almost any domain that uses forms. See Section 8 for additional discussion of strengths and weaknesses of this approach.

Table 1: Glossary of Notation

Symbol	Description
F	The set of forms
D	The set of dictionaries
E	The set of extractor models
m	An extractor model
i	A request (a string)
l	A label (a named substring of a request)
m(i)	The set of labels generated by applying *m* to *i*
T	The set of triples *(i,m,l)* generated by *E(i)*

VIO communicates with a form system based on a relational-database backend. Thus, VIO issues queries on the data and metadata of the form system. VIO analyzes four types of information: the schema, the list of forms, the set of values of every attribute, and the structure of each form represented as a tuple for every field.

For the schema, VIO uses queries on the metadata of the database to extract the set of relations, attributes, and types of the schema. VIO uses the set of all relation-attribute pairs to structure its analysis of requests and to structure its reference resolution.

Each form is associated with a class in a *k*-way classifier. There are *k* forms and $|F| = k$. (Table 1 lists the notation used in this paper.) This classifier is used to generate a ranked list of possible forms, as described in Section 3.

For every relation-attribute pair in the schema that has a string type, VIO reads the set of all attribute values and constructs a dictionary containing all tokens with character lengths greater than 1. Single character tokens are removed because they are not very specific. The result is a set of dictionaries *D*. These dictionaries are used as part of form field analysis, as described in Section 4.

For reference resolution, VIO resolves references to the primary key values of relations that contain entities, as described in Section 5.

3. Generation of Ranked List of Forms

This section describes the learning of a ranked list of forms from weak labels. The domestication of wild labels into weak labels is discussed in Section 6. Sections 7 and 8 describe the experiments that determined the best learning algorithm for this problem.

3.1 Training

VIO trains a *k*-way classifier based on the weak labels to generate a ranked list of possible forms. The feature set used by the classifier is a simple bag-of-words representation of the request. The classifier learns one model per form in *F* that tries to classify an instance as either belonging to that form's class or not. In this paper, four algorithms are evaluated for learning this model: Decision Trees, AdaBoost Decision Trees, Maximum Entropy, and Support Vector Machines [6].

3.2 Application

Each new request is presented to the classifier models. The models generate a score for each form in *F*. The scores correspond to the classifier's prediction for how likely it is for each form to be used to satisfy the request based on the history of requests it was trained on. The forms are ordered by these scores and a threshold function decides how many of the top results to suggest. As noted above, these top-ranked suggestions (if any) are embedded directly into the request as links that will take the assistant to the appropriate form. In addition, the request is augmented with a list of all possible forms to address situations where the correct form is not in the ranked suggestion list.

4. Form-Field Analysis

This section describes the learning of form-field suggestions from weak labels.

4.1 Training

For each relation-attribute pair, VIO trains an annotator using the history of requests and a set of weak labels associated with it. (A weak label in this case is a named substring of a request.) The result of training all extractors is the set of extraction models *E*.

Note that VIO converts annotations to extractions by simply selecting the annotated text. This equivalence of annotations to extractions has a broad set of consequences.

VIO currently uses a conditional random-field [20] extractor with a Begin-Continue-End-Unique internal structure to train the extraction models. The token-level features for the extractors

includes all the words themselves, patterns of case (e.g., MacDonald becomes feature A+a+A+a+), numbers (1234 becomes feature 9+), and punctuation (each punctuation character is separate feature). There are a few features added based on document whitespace and every dictionary in *D* generates a feature if it contains the token. Note that all these features are domain independent.

4.2 Application

Upon receiving a new request *i*, all extraction models in *E* are applied to *i*. When an extraction model *m* is applied to a request *i*, the result is a set *m(i)* of new labels that the model computes as likely examples. The result of applying all the models is a set of tuples that describes the new labels of *i*. The set of tuples *T* is defined as {*(i, l, m) | i is a request, l in m(i), m in E*}. VIO's form-field suggestions are stored in a relation containing relation-attribute-suggestion triples.

When the assistant visits a form to complete the request *i*, the system will suggest the new value in every field on that form for which there is a tuple that was generated by the matching extractor *m*.

5. Reference Resolution for Forms

This section describes the learning of target entity instances from weak labels. VIO automatically selects the *target* entity instances for replace and delete forms. In the first step to solving this reference-resolution problem, the value of every label contained in the set of extracted tuples is checked against the existing values in the database using a string soft-matching process using a Smith-Waterman edit distance, normalized to be between 0.0 and 1.0 by dividing the edit distance by 2 * length of the longer string [18]. Tuples with matches that score higher than 0.8 are retained. A single extraction can produce many matches.

In the second step, the matches are grouped by entity-instance identifier to form feature-vector instances that contain the entity, the primary key value of the entity, and the soft-matching processing weights. For example, the feature-vector instance "person #14, firstname=0.85, lastname=1.0, email=1.0" records the "distance" between the request and a particular tuple #14 in the person relation. These feature-vector instances form a list of examples that might be referenced in the request. To select the correct example, we describe an experiment below where four main methods are tried: the hand-selected field approach, the flat-sum approach, the trained classifier approach, and the IDF approach.

The hand-selected fields approach involves manually selecting a few fields for each entity type that are the best identifiers for a record and, using only those, considering all others to be irrelevant. A weight of 1.0 is assigned to each of the selected fields and a weight of 0.0 to all others. Any candidate entity that matches any of the hand-selected fields will have a non-zero score. Entity instances are again sorted by their overall score. For instance, first and last name were used as the primary fields for person records. This approach introduces a very simple notion of which fields are important to this analysis and which fields are irrelevant.

The flat-sum approach takes the sum of the string soft-match weights for all fields of each candidate entity as its score. Items above a certain threshold are then suggested, sorted by score. This approach treats all fields as being equally important in selecting the correct record.

The trained-classifier approach trains a Naïve Bayes linear classifier on a corpus of labeled examples. The classifier model is then used to evaluate each new example and attach a score to it. As with the previous methods, the scores are sorted and compared to a threshold value to determine which records to suggest. This approach learns weights to express the relative importance of different attributes in identifying entities that are referenced.

Finally, the inverse-frequency-weighted sum is calculated by weighting each value based on how many matches the extracted value had instead of which field it is. Generation of this weight is treated much the same as an IDF weight that might be used in an Information Retrieval system. The formula log (corpus size/document frequency) provides the weight with the total number of tuples of that type being used as the corpus size and the number of tuples that matched any given value filling in as the document frequency. This approach suppresses the importance of extractions that produce many matches. For example, the fact that the title of "Researcher" matches for a person means less if that same title also matches many other people. The match of a unique email address is very strong evidence. This approach does not learn the relative importance of different attributes, only the relative importance of specific values based on each value's distribution.

Note that, because it is based on the set of extractions, performance for reference resolution is bounded from above by the recall of the extractors. If the extractor missed values, then the entity recognition cannot identify the correct instance associated with the missed values. To counter this performance penalty, we took all the extractors in the system and "tweaked" them so that the models generated higher recall but lower precision [17]. The tweaked extractors were used for reference resolution and not for pre-filling form fields. The internal bias term was re-weighted to maximize the extractor's F3 score based on its own training data. These new extractors were then used to generate the inputs to the four reference resolution algorithms, thus generating four more methods. The results of the eight different methods are presented in Section 8.

5.1 Application

After scoring the list of candidate instances, sorting them, and discarding instances with scores below the threshold, the final list is stored in the database. The list is used to automatically select an instance whenever an assistant is required to do so. For example, in a request to update information about a person, the assistant must select an existing person as the target for a modification. A dropdown list on the form automatically selects the top-ranked suggestion, and all the suggested values in that final list are listed in rank order at the top of the dropdown list.

6. Generation of Training Data from Assistant Interaction

As an assistant interacts with VIO to process requests, the messages that specify the request are logged along with the actions that are taken by the assistant in the form system to satisfy that request. This fact requires the assistant to have originally navigated to the forms using the embedded link in the request. This link back to the original request is maintained even if the

assistant changes forms or edits values. The log includes the full text of the original request, the identifier of the form that was used to process the request, the entity instance that was inserted, deleted, or modified, and a set of triples that record all field values (similar to the ones that are generated by extraction). These triples form the set of "wild labels" *W* for a request. These wild labels consist of a request identifier *i*, an extraction model *m*, and a label value *l*, which is a text string taken from the field values on the form. There is one wild label per text value appearing in any field on the form.

All wild labels are gathered purely by observation and are used to generate the training data without additional feedback from the assistant or the need for a human expert to do any hand labeling or corrections of any values.

6.1 Labels for Form Ranking

The training data for the classifiers consists of a set of base requests and a single weak class label for each classifier. The text of the original requests is taken directly from the logs to form the examples for training. The identifier of the form that was used to complete each request is taken from the form log and is applied to the corresponding request as its label. Thus, each request receives a weak label for exactly one class.

6.2 Labels for Form Field Analysis

The training labels needed by the extractors consist of a set of request examples and labeled *spans* within those requests corresponding to the set of extraction models. Again, the texts of the original requests are used to generate the set of training examples. The wild labels are then used to generate the labeled spans. However, wild labels consist of string values taken from the forms and the training examples are named spans of text in the request.

In order to convert the wild labels into the necessary training labels, the *domestication* algorithm calculates Smith-Waterman string-edit distances between every wild label and a moving window of tokens in the base message. A set of potential matches is generated by filtering those string-edit-distance scores against a similarity threshold, and a final pass drops duplicate labels and chooses the best scoring match. To choose the best scoring match, both scores within a label type and between labels are considered. So, for example, the first name label of "Leah" will not match the string in the e-mail label of "leah.davies@cs.cmu.edu" if an e-mail label has a better match. The set of named spans identified by this process form the weak labels, and together with the base requests make up the training data set used by the extractors.

Not every wild label necessarily generates a weak label. Many values appear that, on the form and in the wild-label set, are not specified in the original request. At the same time, a single wild label may generate more than one weak label in a request.

For example, the request "*Leah Davies' phone number is now 555-1212*" can be completed using a modify-person form. This form generates a wild-label set that include all of Leah Davies' data fields—email, phone, office location, etc. But only the first name, last name, and phone number appear in the body of the request.

The string edit soft-matching catches simple typos or formatting changes, but it will not properly generate domesticated labels when a more complex action was taken to translate the value in the request to the final value that was entered on the form. For instance, if a request referred to "Next Monday" but the assistant entered "4/17/06" into the form or a reference to "Dave's Office" was entered as "252 Baker Hall," the system would be unable to label either of those spans in the original request because it has no deeper understanding of the values and is simply working with string-edit distances. However, our experimental results show that this type of language is rare in information intent requests.

The domestication algorithm makes the fundamental assumption that the wild-label set contains every instance of an attribute that appears in the request. This assumption is sometimes violated and the weak-label set will be missing some labels.

For example, the request "*... my email address is incorrect on the Ardra contact info page. It's listed as: bcruise@ardra.org, but it should be bradcruise@ardra.org. Perhaps they got it confused with bcruise@fairbox.com ...*" contains three variations of one e-mail address. The form used to complete this request will have an initial value of *bcruise@ardra.org* in the email field and a final value of *bradcruise@ardra.org*. Both of these values are in the wild-label set *W*. However, *bcruise@fairbox.com* appears nowhere on the form and as such will not be in *W*, and not be labeled as an email address in the domesticated labels.

The assignment of one extractor for each attribute also causes confusion in training. For example, the cell-phone extractor is different than the home-phone extractor. A request that mentions cell-phones and home-phones will have a weak cell-phone label and a weak home-phone label. But the cell-phone label will be a *negative* example for the home-phone extractor and visa-versa. The extractors will struggle with similar label types when it does not have enough information to differentiate between them. This "split-label" problem is discussed in Section 8.6.

6.3 Labels for Reference Resolution

The weak label for reference resolution consists of the selection of the instance of the principle entity modified by a particular form. In the case that the form is completed correctly, the weak label is exactly equal to the golden label. For this paper, every form was completed correctly by the assistant, so we do not distinguish between the two sets. (However, poor interaction design can cause problems, see [7] for more details.)

7. Experimental Framework

To better understand the analysis of information intent, domestication, and the performance of machine learning on weak labels, we performed a variety of experiments. Domestication performance is measured by comparing the results of domestication to a gold-label set. Form-ranking performance, form-field performance, and reference-resolution performance are measured over time to track performance improvement as assistants continue to use the system. Overall system performance of information-intent analysis is measured on a final test set.

To ensure that the system's evaluation is based on a realistic workload, two collections of requests were gathered from existing real-world sources. For the *seminar corpus*, several Carnegie Mellon University mailing lists were archived. From the archive, 439 seminar-announcement messages, which announced a single seminar, were extracted. Each message was a request for attendance at a seminar talk. From this corpus, 60 of these

requests were set aside for a testing set and the remainder formed the training set. All requests were processed by first presenting each request to an assistant. A single assistant filled out a form that added the appropriate data of the announced seminar into a database. All requests were additionally hand labeled to create a gold-label set for extraction. This corpus was only used to evaluate domestication and extraction performance.

For the *webmaster corpus*, a set of e-mail requests was extracted from a log of saved messages of a professional webmaster. The website update requests in this log were processed with the following steps. The requests were first anonymized by translating real-world entities into parallel fictional entities of a made-up website. Then requests were stripped of thread replies. Messages containing multiple requests were split into multiple messages. Finally some requests were discarded because the requests could not be satisfied through the completion of a form. All these transformations preserved the *language* of the original requests as much as possible. The result was 234 requests. Chronologically, the first 195 requests were used as a training set and the last 39 requests were used as a test set. The test set was used to measure the performance of VIO after a short "warm-up" phase of interaction that allows the machine learning algorithms a chance to learn.

All requests were presented to an assistant who completed each using VIO. In addition, all requests were also hand labeled with a gold standard for extraction. Since every request was processed correctly, the log of form and entity-instance selections generated by the assistant served as the gold label sets for those algorithms.

While the implementation of VIO on our made-up website allowed assistants to make 18 different requests types, the e-mails in the webmaster corpus covered a narrower set, involving only updates to information related to people, sponsors, and news information. Thus, the use of real-world data introduced a realistic skew into the distribution of request types.

All of the training data used in experiments were gathered automatically by processing the training portion of the requests in an untrained version of the system. Learned models were trained at five approximately evenly spaced points in time during this process to study performance improvement over time.

Domestication performance is reported in terms of precision, recall, and the harmonic mean (F1) for each extractor type for an exact match of the entire label ("entity" precision). The scores are generated by comparing the domesticated labels to a set of gold labels for the requests in the training set.

Form-ranking accuracy is reported in terms of the Mean Reciprocal Rank of the correct form in the ranked list of suggestions. Classifiers were trained using the requests from varying amounts of the training set.

Extractor performance is reported as precision, recall, and the harmonic mean (F1) value for each extractor separately. Extractors are trained on varying numbers of requests from the training set using the domesticated labels. The resulting models are then applied to the requests in the test set and the resulting extractions are compared to the test set gold labels.

Reference-resolution performance is reported only for person-recognition requests. There were not enough reference-resolution examples for other entity types in our test data to produce meaningful measurements. Only modify and delete requests are considered since add-person requests do not require the assistant to select an existing record. Within this subset of the requests, the Mean Reciprocal Rank of the correct person is reported. Experiments were run using the output of extractors trained on various amounts of training data, in particular depending on the performance of extraction at the corresponding point in time. In addition, tweaked versions of these extractors using all four weighting schemes for scoring are reported.

8. Experimental Results

In this section the experimental results of two corpora are described. One corpus covers all steps in the analysis of information intent. An additional corpus provides additional results for our domestication and extraction algorithms.

8.1 Domestication Performance

Table 2: Domestication Performance on Seminar Corpus

Label Name	Precision	Recall	F1
person_name	0.8649	0.9432	0.9024
seminar_day	0.9437	0.9860	0.9644
seminar_month	0.9086	0.9915	0.9482
seminar_year	0.8989	0.9826	0.9389
seminar_title	0.8433	0.9022	0.8718
seminar_starttime	0.9616	0.9921	0.9766
seminar_endtime	0.9839	0.9892	0.9865
seminar_building	0.9605	0.9884	0.9743
seminar_room	0.9529	0.9582	0.9555
Micro-Averaged Values	0.9207	0.9679	0.9435
Macro-Averaged Values	0.9243	0.9704	0.9465

Table 3: Domestication Performance on Webmaster Corpus

Label Name	Precision	Recall	F1
person_bio	1.0000	1.0000	1.0000
person_email	0.9759	0.9759	0.9759
person_fax_phone	0.7778	0.7778	0.7778
person_first	1.0000	0.9340	0.9659
person_last	0.9671	0.9484	0.9577
person_office_phone	0.7447	0.7778	0.7609
person_organization	0.9818	0.9474	0.9643
person_org_city	1.0000	0.7778	0.8750
person_org_department	1.0000	0.9091	0.9524
person_org_url	1.0000	1.0000	1.0000
person_org_location	1.0000	1.0000	1.0000
person_org_state	1.0000	0.8889	0.9412
person_org_street	1.0000	1.0000	1.0000
person_org_zip	1.0000	0.8000	0.8889
person_personal_url	0.9545	0.8571	0.9032
person_title	0.9853	0.9306	0.9571
sponsor_name	1.0000	1.0000	1.0000
news_body	0.7500	0.6923	0.7200
news_header	1.0000	1.0000	1.0000
news_url	1.0000	1.0000	1.0000
Micro-Averaged Values	0.9660	0.9276	0.9459
Macro-Averaged Values	0.9569	0.9109	0.9320

Table 2 lists the performance of the domestication algorithm for every extractor in the seminar corpus. In general, domestication performs well with good precision and very high recall. For name

and year, precision is lower because the algorithm finds all occurrence of a name or year, but only the reference in the announcement header was labeled as "gold." For the title label, this duplicate reference problem also occurred, but title additionally had difficulty with plain-text formatting tricks used to format the presentation of the seminar request. In general recall is high because copy-and-paste was used to complete the form.

Table 3 lists the performance of the domestication algorithm for every extractor in the webmaster corpus. (The label bio stands for biographical sketch, generally a one-line description of hobbies. The label org stands for organization.) The number of label instances in the training set is listed in Table 5. The domestication algorithm performs very well, except for fax and office numbers, and for the body of requests to update news on the website. Phone numbers are difficult because a request string such as "412.555.1212" is translated into "(412) 555 1212" due to embedded formatting issues in the form system. Domestication of news body sometimes failed because of the maximum window size for domestication analysis.

8.2 Extraction Performance

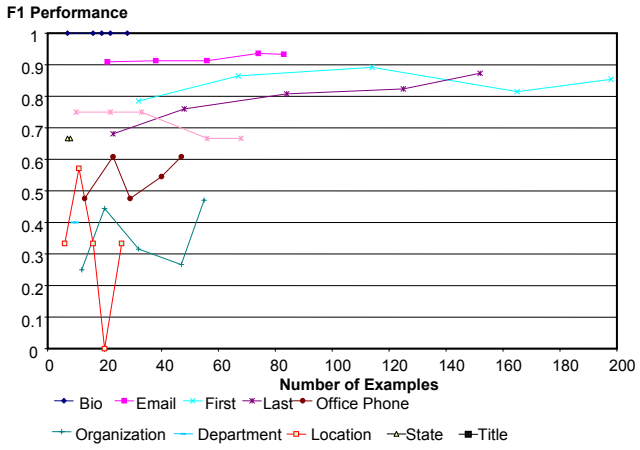

Figure 2: Extraction Performance

Figure 2 is a graph of extraction performance over time on the webmaster corpus. The x-axis in this graph is the number of instances of that particular label (not the total number of requests). As described earlier, the y-axis is the performance of the extraction model on the test set after training on the given number of examples. The ten lines indicate the extractors that generated non-zero extractions. Many extractors had insufficient data to generate extractions.

To interpret Figure 2, when VIO is first used, it behaves identically to a direct-manipulation content-management system that has no intelligence. Extraction of true positives help the assistant by pre-filling the form. A false negative of any extractor has *no* negative effect on the assistant. A false negative is simply a missed opportunity to help the assistant. (See reference [7] for a description of the evaluation of the assistant experience.) When an extractor generates a false positive, the assistant must correct the mistake. (Table 5 below shows that the extractors are biased towards precision.) Figure 2 also demonstrates that extractor performance generally improves with additional examples. The erratic performance of some extractors is a consequence of three factors: the small test-set size for the extractor, the small number of examples, and errors in weak labels. The extractors that occur often (i.e., extend to the right) perform well. The rarely occurring extractors have less of an effect simply because the corresponding "problem" of completing the corresponding form field does not often occur.

Table 4 lists the performance of extraction on the seminar corpus. In this experiment there are two principle effects. One is the consequence of a single label that is considered correct in the gold-label set. The second effect, however, is directly related to the domestication algorithm itself. In many requests day, year, start time, and end time are entered into the corresponding form as a single digit or pair of digits. This form of data entry has two effects. First, every occurrence of the digit or digits in the corpus is labeled by the domestication algorithm. Second, the extraction algorithm tends to label every occurrence of those digits. To address this issue, we plan to add context to the domestication to improve its accuracy.

Table 4: Extraction Performance on Seminar Corpus

Label Name	#	Precision	Recall	F1
seminar_building	354	0.8545	0.8246	0.8393
seminar_room	361	0.9434	0.8475	0.8929
seminar_month	383	0.9107	0.8095	0.8571
seminar_day	373	0.8226	0.7846	0.8032
seminar_year	188	0.8077	0.7000	0.7500
seminar_starttime	391	1.0000	0.6333	0.7755
seminar_endtime	186	1.0000	0.5000	0.6667
seminar_title	383	0.7692	0.1667	0.2740
person_name	422	0.6515	0.6143	0.6324

Table 5: Extraction Performance for Webmaster Corpus

Label	#	Precision	Recall	F1
person_bio	28	1.0000	1.0000	1.0000
person_email	83	1.0000	0.8750	0.9333
person_fax_phone	9	0.0000	0.0000	0.0000
person_first	198	0.9722	0.7609	0.8537
person_last	152	0.8889	0.8571	0.8727
person_office_phone	47	1.0000	0.4375	0.6087
person_organization	55	1.0000	0.3077	0.4706
person_org_city	7	0.0000	0.0000	0.0000
person_org_department	10	0.5000	0.3333	0.4000
person_org_url	8	0.0000	0.0000	0.0000
person_org_location	26	1.0000	0.2000	0.3333
person_org_state	8	1.0000	0.5000	0.6667
person_personal_url	44	0.0000	0.0000	0.0000
person_title	68	1.0000	0.5000	0.6667
sponsor_name	5	0.0000	0.0000	0.0000
news_header	15	0.0000	0.0000	0.0000
news_url	13	0.0000	0.0000	0.0000
news_body	12	0.0000	0.0000	0.0000

Table 5 reports the performance of all extractors trained on all 195 training requests of the webmaster corpus. As noted before, entity precision, entity recall, and entity F1 measure the model's predictions against the gold-labeled test set. These results illustrate that, once about 30 examples or so are available, the extractor generates a high-quality result. The principle exceptions to this rule are the URL extractors. These extractors perform poorly because the single conceptual class of a URL is split into three classes: personal-organization URLs, personal URLs, and

news URLs. Thus, URLs are an example of the split-label problem. Each extractor in isolation never has sufficient evidence to ever extract a URL. The benefit of the automatically generated dictionaries has a positive impact on many fields. The person_org_state extractor performs well because the dictionary already contains all the state names used in the test requests.

8.3 Form-Ranking Performance

Figure 3 is a graph of the form-ranking performance over time for the webmaster corpus. All the algorithms tested improved over time. AdaBoosted Decision Tree and Maximum Entropy perform consistently better than Decision Tree and SVM. However, without more data, it is not clear which algorithm is best. Note that after full training, the correct form appears in the top two of the ranked list of forms, except in one test example.

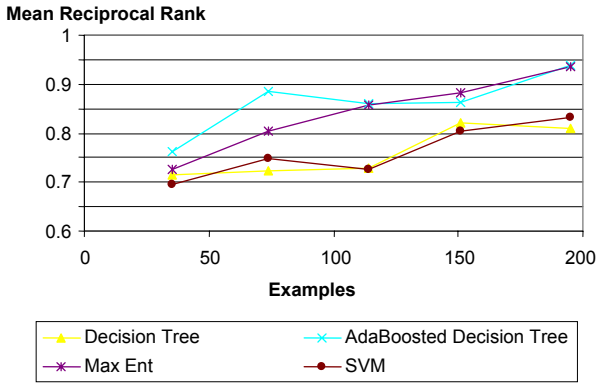

Figure 3: Form Ranking Performance

8.4 Reference Resolution Performance

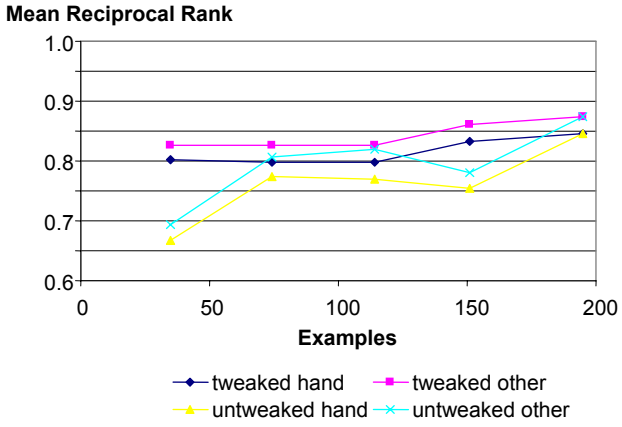

Figure 4: Reference-Resolution Performance

Of the four approaches to weighting dictionary matches, three of them—the flat-sum approach, the trained-classifier approach, and the IDF approach—performed equally well across every condition. These three approaches are reported together as a single "other" value along with the hand-selected field approach. Additionally, the number of training examples is varied and both tweaked and un-tweaked extractors were used to generate the inputs. Figure 4 is a graph of the performance of reference resolution on the webmaster corpus over time. The graph shows that the tweaked three "other" approaches consistent outperform the hand-selected approach in initial performance. However, additional data might reveal that all the approaches converge or that the "other" approaches separate.

Reference-resolution benefits from using all of the fields to identify requests are systematically better than the hand-selected approach. Additionally, tweaked versions of the extractors provide a large benefit to the system when it does not yet have enough training data, though this margin shrinks as the extractors improve and we are less able to squeeze out extra recall.

8.5 Overall Performance

Overall the system performs quite well in this domain of problems. It is quite reliable at suggesting the correct form as well as picking the top-ranking entity instance for modification. Extractions perform well for short common fields, but poorly on long values and uncommonly used fields. Since performance is related to the frequency that a field appears, good performance appears where automation offers the largest time savings. Additionally, precision remains generally high in all cases, and as such doesn't actually hinder the assistant by suggesting incorrect values.

Our tests were performed on a training set with a relatively limited number of requests. Performance picks up quickly, but would continue to improve in an actual deployed system, in which we could expect to see many more requests. This fact is particularly true of the extractors for relatively uncommon fields and classifier accuracy on infrequently used forms. In both of these cases, the system suffers from a lack of sufficient training data.

8.6 Discussion

VIO uses a simple "equality" model of conversion between annotations and assistant interaction. In the reserves direction, domestication also assumes a simple mapping model between the wild label and the weak label. This model works perfectly for form selection and reasonably well for entity-instance selection. For extraction, annotations are directly converted to extractions. Domestication converts text-field values into weak labels relatively directly. In addition to handling text fields, VIO can be easily extended to handle pull-down menus of a fixed set of choices, "radio" buttons, sliders, etc., where the bi-directional translation between the widgets and the annotation is accurate. For experimental purposes, we stuck to this model to minimize the addition of domain-specific knowledge. However, more complex transformations require a more sophisticated, domain dependent, mapping. For example, consider a calendar widget for date of birth that should be set to June 13th, 1961. The annotation label in a message might be "6/13/1961," or "June 13 1961," or even "my birthday." To handle conversions that require some domain semantics, we plan to imbed semantics directly into the widget itself, essentially as part of "widget authoring." This approach leverages the creation of a plug-in library of domain-specific widgets that can easily be shared across applications and organizations.

Experimental results show that the domestication algorithm generally works well, but does not handle the one case where additional labels, not appearing in the form, should be labeled as part of a particular class. For example, many e-mail signatures

contain a person's name, title, cell-phone number, etc. But these strings never appear in the form request and thus are never labeled with the associated class by domestication. Ironically, this situation improves performance for our webmaster corpus, because the signature is frequently from an individual who is *not* the target of the request. Thus, for example, the extraction algorithm quickly learns not to extract "Ashuk," (the pseudonym of a person that frequently composed requests to the webmaster). While this particular case worked out well in the experimental corpus, we believe that deeper semantic analysis of requests, e.g., signature recognition, would benefit VIO.

Another consequence of the goal of domain independence is VIO one-to-one association of form fields with extractors (since no additional information is needed to make this association). This association leads to a split of one "conceptual" label and extractor to several actual labels and extractors, thus killing performance in some cases. A simple one-level indirection extraction results to form fields may be a sufficient solution for this problem.

9. Related Work

Command-style natural-language systems related to information intent have a scattered history, beginning with SHURDLU [15] and ASK [11] and continuing to the work of Mooney and his students [3].

The ASK system 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 handling of errors of any type.

Mooney's work [16] is similar because of the emphasis on semantic parsing. A particular advantage of this work is the emphasis on a context-free target language. Our target language consists of a combination of attribute/value pairs, forms (which can be thought of as "templates" or frames with slots), and references to entities. On the other hand, VIO handles new values.

In our own work, reference [1] reports on some initial experiments in the analysis of website requests using gold-labeled machine-learning techniques. Reference [2] reports on experiments that measure the performance advantages of information-intent analysis in a cooperative work environment. Reference [7] reports on the assistant experience and results of behavioral testing of VIO. VIO uses MinorThird [6] implementations for all its machine learning.

Horvitz [8] describes a system with a similar interaction model that focuses on a single domain of scheduling meetings into a calendar system. This work uses gold labels and concentrates on agent confidence, expected utility, and the problem of initiation of dialog with the assistant (in a non-irritating way).

The MANGROVE project [10] is targeted at bridging the chasm between the message-unstructured world and the database-structured world. This project implements a direct-manipulation-interaction-style markup tool for messages. Our prototype may be applicable to the same scenario.

Lockerd, et. al. [5] describes a user study of e-mail-based requests to a web master for changes to a website. 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 research, e.g., [8][13]. VIO represents the next step in this line of research where weak labeling is used instead of gold labeling.

Meng [12] 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 references via a weighted n-gram vector cosine function. This work is related to our work on extraction.

Finally, a variety of works, for example, [9][10][22][23][24], focus on extending e-mail with semantics or task structure. VIO is clearly complementary to this line of research.

10. Conclusion

People frequently send messages containing *information intent:* requests that specify a task to be accomplished by an assistant. In this paper we proposed analyzing information intent by using machine-learning algorithms trained on weak labels. Weak labels are generated by passive observation of the assistant's interaction with the system. The results of the analysis are used to aid the assistant in processing the request through an *intelligent form interface*.

To better understand the performance, advantages, and disadvantages of our solution, we built an end-to-end prototype, the Virtual Information Officer (VIO), which operates by analyzing the information intent of the user, providing an interface for the assistant to inspect the results of analysis via a form-based data-entry system, executing the user's intent as updates to a database, and generating weak labels from the interaction. Weak labels are used to re-train models to automatically improve performance over time.

In addition, the prototype was build with a minimum of domain specific engineering to ensure that our intelligent interface can be retargeted to new domains with a minimum of engineering effort.

We then used this prototype to perform a series of experiments to determine the performance of the system and to determine the best machine-learning algorithms for analysis. Our experiments showed that our algorithm for generating weak labels, termed *domestication*, performed very well measured against a hand-generated "gold label" standard. This result is crucial to the success of the system because weak labels provide the training input to the rest of the analysis.

For information-intent analysis, three problems were solved: form ranking, form field pre-filling, and entity reference resolution. Our experiments showed that AdaBoosted decision trees, which beat out three other machine-learning algorithms, performs very well. On our final test set, the correct form was suggested either first or second in every case but one. For form-field pre-filling, we used conditional random field (CRF) extraction. CRFs worked well as long as the weak labels were accurate, except in the "split label case," i.e., when a label for a general category of sequence (telephone number) was split across two classes (cell-phone verses home-phone). This split confuses the extractor since each

positive cell-phone label is a negative example for a home-phone label, and visa versa. For reference resolution, we used a reference resolution algorithm based on learning important feature *differences* between features present in the information-intent request and features present in entity-reference instances. The algorithm performed as well as two other competitors, given the limitation set by the extractor performance. Tweaking [17] the extractors (boosting recall at the expense of precision) has more of an impact when their initial performance was low.

In summary, our analysis shows that information intent can be successfully analyzed using weak labels as training evidence. In related work [7], we show that the user interaction supported by the analysis significantly improves assistant time performance without a significant introduction of new errors.

11. Acknowledgements

Thanks to Robert McGuire and Kelvin Goh for their software engineering work on VIO. Thanks to Paul Bennett for an insightful critique of our work. Thanks to Emily Leathers and Ken Mohnkern for assistance in the project. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHC030029.

12. REFERENCES

[1] Cohen, W. W., Minkov, E., Tomasic, A., Learning to Understand Web Site Update Requests. In *Proceedings of IJCAI* (2005), 1028-1033.

[2] Tomasic, A., Zimmerman, J., Simmons, I., Linking Messages and Form Requests. In *Intelligent User Interfaces (IUI)* (2006), 78-85.

[3] Mooney, R., Learning Semantic Parsers: An important but under-studied problem. In *Working notes of the AAAI spring symposium on language learning* (2004).

[4] Tomasic, A., Cohen, W., Fussell, S., Zimmerman, J., Kobayashi, M., Minkov, E., Halstead, N., Mosur, R., and Hum, J., Learning to Navigate Web Forms. In *Workshop on Information Integration on the Web (IIWEB)* (2004).

[5] Lockerd, A., Pham, H., Sharon, T., and Selker, T, Mr.Web: An Automated Interactive Webmaster. In *Proceedings of the Conference on Computer/Human Interaction (CHI),* ACM Press (2003).

[6] Cohen, W., *Minorthird: Methods for Identifying Names and Ontological Relations in Text using Heuristics for Inducing Regularities from Data*. http://minorthird.sourceforge.net.

[7] Zimmerman, J., Tomasic, A., Simmons, I., Hargraves, I., Mohnkern, K., Cornwell, J., McGuire, R., VIO: A mixed-initiative approach to learning and automating procedural update tasks. In *Proceedings of the Conference on Computer/Human Interaction (CHI),* ACM Press (2007).

[8] Horvitz., Principles of Mixed-Initiative User Interfaces. In *Proceedings of Conference on Human Factors in Computing Systems (CHI),* ACM Press (1999).

[9] Bellotti, V., Ducheneaut, N., Howard, M., Smith, I., Taking Email to Task: The Design and Evaluation of a Task Management Centered E-Mail Tool. In *Proceedings of Conference on Computer/Human Interaction (CHI),* ACM Press (2003), 345-352.

[10] Etzioni, O., Halevy, A., Levy, H., and McDowell, L., Semantic Email: Adding Lightweight Data Manipulation Capabilities to the Email Habitat. In *International Workshop on the Web and Databases (WebDB), June 12-13, 2003, San Diego, California* (2003).

[11] Halevy, A., Etzioni, O., Doan, A., Ives, Z., Madhavan, J., McDowell, L., Tatarinov, I., Crossing the Structure Chasm. In *Conference on Innovated Data Systems Research (CIDR)* (2003).

[12] Thompson, B., Thompson, F., Introducing ASK, A Simple Knowledgeable System. In *Proceedings of the First Conference on Applied Natural Language Processing,* Santa Monica, California (1983).

[13] Meng, F., A Natural Language Interface for Information Retrieval from Forms on the World Wide Web. In *Proceedings of the 20th International Conference on Information Systems,* Charlotte, North Carolina (1999).

[14] Cohen, P., et. al., Synergistic Use of Direct Manipulation and Natural Language. In *Proceedings of the Conference on Computer/Human Interaction (CHI),* ACM Press (1989).

[15] Cohen, W., Ravikumar, P., Fienberg, S., A Comparison of String Distance Metrics for Name-Matching Tasks. In *Workshop on Information Integration on the Web (IIWEB)* (2003).

[16] Winograd, T., *Understanding Natural Language,* Ph.D. thesis, Academic Press (1972).

[17] Kate, R., Wong, Y.W., Mooney, R. J., Learning to Transform Natural to Formal Languages. In *Proceedings of AAAI* (2005).

[18] Minkov, E., Wang, R. C., Tomasic, A., Cohen, W. W., NER Systems that Suit User's Preferences: Adjusting the Recall-Precision Trade-off for Entity Extraction. In *HLT/NAACL* (2006).

[19] SecondString, http://secondstring.sourceforge.net/.

[20] Lafferty, J., McCallum, A., Pereira, F., Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of 18th International Conf. on Machine Learning (ICML)* (2001).

[21] Schapire, R. E., Singer, Y., Improved Boosting Algorithms using Confidence-rated Predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (1998), 80–91.

[22] Cohen, W., Carvalho, V., and Mitchell, T., Learning to Classify Email into “Speech acts.” In *Conference Empirical Methods in Natural Language Processing* (2004).

[23] Dredze, M., Lau, T., Kushmerick, N., Automatically Classifying Emails into Activities. In *Intelligent User Interfaces (IUI)* (2006).

[24] Shen, J., Li, L., Dietterich, T., Herlocker, J., A Hybrid Learning System for Recognizing User Tasks from Desktop Activities and Email Messages. In *Intelligent User Interfaces (IUI)* (2006).

[25] Felligi, I., Sunter, A., A Theory for Record Linkage. In *Journal of the American Statistical Society*, 64:1183--1210 (1969).