Collective Online Form Schema Integration

Andrés C. Rodríguez
SRI International
333 Ravenswood Avenue, Menlo Park, CA 94025
acr@ai.sri.com

Abstract

Forms are part of our every day experience on the web. Whether we fill forms to travel, register to a newspaper or fill in our timecards, our office day is usually full of form-filling activities. A novel way of discovering and mapping semantic form concepts is proposed by using the data entered by a community of users. As a first step, a dynamic naïve probabilistic model behind is built behind any online form that a member of the community comes across. This model predicts probable values based on the correlation between form fields. As a second step, the data from those models is used to find the probability that a field in one form corresponds to a field in another form. Once that mapping is established the models (and data) can be reused for new website forms. As a side note a method for getting the same results while maintaining users data anonymous is shown.

While there have been multiple attempts at easing the entry of data at the web, most of them focus around some sort of natural language understanding (i.e. understanding the language on the web page). Furthermore, most of the attempts at matching schemas from the "deep web" rely on batch access to the data. The presented algorithm is an online learning algorithm that reinforces or weakens the systems's belief in matching concepts.

1 Introduction

Our interaction with the web consists of basic actions. People can follow links to other pages or submit information using forms. If the web is thought of as a space where information is stored (in whatever format that might be), forms can be construed as the main information entry point. There are of course other ways people modify information to the web (for example clicking to remove a record), but forms are specially important for electronic commerce, customer support and information brokering.

The work this paper describes is aimed at predicting the value that an internet user will enter into a form field on any given website. That is, according to the system's internal models and schema matchings (to be described below), it tries to find a value that has a high probability of being what the user will enter next.

In order to accomplish this goal, two subsystems are devised. The first, will build a probabilistic model based on the correlations between form fields within a single form using a Naïve Bayes formalism. The assumption being that there is a relationship between form fields. For example a person's first and last name should be highly correlated. If two people have the same last name, then other form fields should be able to disambiguate: they live in different zip codes. This subsystem, called the *Single Form Predictor* "listens" to people in the community as they submit forms while they browse the internet, and updates the model. When a prediction is needed, the subsystem will

receive the partially filled form as evidence, calculate the probabilities for the all the known values, and offer and ordered list.

The second subsystem, called *Form Schema Matcher* uses the learnt values for each form field and try to match them against other form fields. For example, while submitting a travel reimbursement request at work, a user fills her first name, last name and address. Then, if the same user goes to a newspaper website, and registers herself, she will use the same values as the ones in the travel reimbursement request. Assuming there are other instances of the match, the schema matcher will reinforce its belief that the field called $name$ in the travel reimbursement form corresponds to the field called $fname$. The names need not be semantically similar, since the driver of the match is the values of the form fields.

By using the connections stemming from the matcher, a form filler can now draw on data from different forms. If data for a website visited for the first time is not available (or the maximum value is still too low), one can look at either the prior probability for the field we are trying to predict, or the conditionalized probability.

2 Related Work

In the field of form filling, there are two well known commercial examples: Google Toolbar, and RoboForm. Those two are examples of a form filler that receives personal information from the user, and then tries to identify use cases for that information in the pages the user visits. For example, Google Toolbar will ask the user for its name, address, telephone and credit card information. then, when visiting an e-commerce website, it will try to match the information it has with what is being asked. The main drawback of this approach is that the set of fields is set from the start. There is no possibility of the system learning other fields that the user enters repeatedly. For example, the employee ID for work related forms.

Most of the work on form filling, as well as the implementations are knowledge centric, in the sense that they rely on knowledge of how the form is structured. A difference between those approaches and the one presented here is the use of knowledge and the application of machine learning techniques. Knowledge heavy approaches rely on knowledge engineers to build the infrastructure.

The field of schema matching has thrived recently with the advent of the web and the many information sources available from it. Most of the literature focuses on finding pairs of related fields among "similar" databases. The goal here is quite different, since the schemas overlap can be very small or non-existent. Among the subdivisions of schema matching, this approach falls under the "Contents Based". Being content based means that it is assumed that the column names are (or can be) opaque.

3 Single Form Predictor

A form is a commonplace object, so much so that depending on who is asked to define it, one might get a completely different answer. An online form has a very precise representation on HTML, but the database community might mention complex forms such as the ones in a master-detail relationship, or conditional forms might be discussed by people at the census bureau.

Figure 1 depicts a simple form with n fields. A specific Field f_i is highlighted.

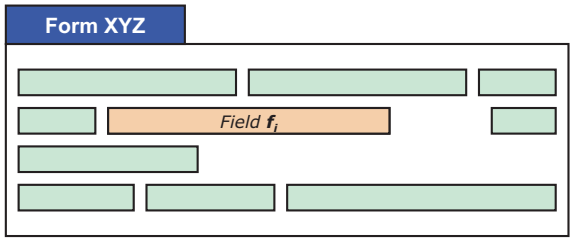

Figure 1. Form with n Fields

3.1 Definitions

To describe the approach, definitions of what it is understood by *form* and by form *fields* are in order. An online **form** F is a collection of form fields:

$$F = \{f_1, f_2, ..., f_n\} \tag{1}$$

A **field** f_i is an element of the form. In the general case, fields can be annotated with types. If no type annotation is present (as is the case with HTML) fields are typed as alphanumeric strings.

A form **instance** F^I is an assignment of values to each one of the form fields. The values can possible be empty (designed as λ).

$$F^I = \{v_1^I, v_2^I, ..., v_n^I\} \tag{2}$$

We define (historical) **data** $\mathcal{D}$ to be a collection of instances,

$$\mathcal{D} = \{F^1, F^2, ..., F^M\} \tag{3}$$

The **context** $\mathcal{C}$ in a form predictor is a collection of facts that are known to it. Context is a somewhat ill-defined concept, in the sense that it can be as simple as the current time and as complex as the collection of all documents stored in the user's computer. The objective of including it in the definition of the predictor is to make that definition as general as possible. A better example of context is the HTML interweaving the markup that constitutes the form.

A form filling **strategy** $\mathcal{S}(F)$ for a form F is a set of functions

$$\mathcal{S}(F) = \{S_1, S_2, ..., S_n\} \tag{4}$$

such that each S_i is of the form

$$S_i : \mathcal{C}, \mathcal{D}, F^{new} \rightarrow V(f_i) \tag{5}$$

Where F^{new} corresponds to the instance assignment of the partially filled form (for example if the user is half way in a 10 field form, then it will contain a partial assignment with the first 5 values). $V(f_i)$ is the set of possible values the field can take (including λ).

Two common used strategies when filling forms are to use the *most recently used* (MRU) value, and the *most frequently used* (MFU) value.

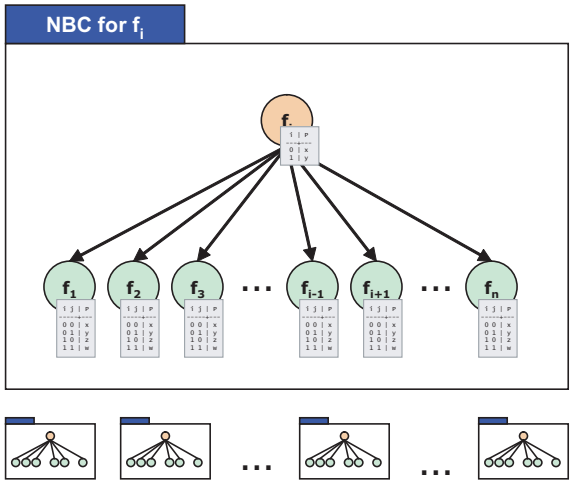

Figure 2. naïve bayes Classifier for field f_i

3.2 Model

The main purpose of this sub section is to define the structure of a strategy that uses a collection of naïve bayes models as form field predictors. The assumption are the following:

- Fields f_i behave as random variables
- The domain of f_i as a random variable is $V(f_i)$
- When trying to predict field f_i, this field "causes" other fields
- Fields f_j and f_k are independent of each other given f_i

Based on these assumptions, n naïve bayes classifiers (NBC) are constructed for each form, that is one per field. A naïve bayes classifier is normally used in a setting where one must find the most probable class for an entity, given certain feature space values. The analogy for the current application goes as follows: the seen values for the form are the classes we are trying to distinguish from, the feature space are the other fields in the form.

Figure 2 shows a naïve bayes classifier for a field f_i. Note that all the fields in the form (except for the one we are trying to predict, f_i) make the feature space. The **root node** corresponds to the field we are trying to predict and the **leaf nodes** to the other fields in the form. The figure is also showing the conditional probability tables (CPT) associated with each node in this type of classifiers. The CPT's will be calculated online using maximum likelihood. As new data arrives (from a form submission) the CPT's for each form field, and for each form field NBC are updated. The strategy $\mathcal{S}$ is going to be dictated by each $S_i = NBC(f_i)$ where by the Naïve Bayes classification rule:

$$NBC(f_i) = \underset{v_k \in V(f_i)}{\operatorname{argmax}} P(F_i = v_k) \prod_{j \neq i} P(F_j | F_i = v_k) \tag{6}$$

Additionally to the form fields there is an artificial *identity*. This field will be artificially added to each form and automatically filled with an Universally Unique Identifier (UUID) that is assigned to each user. Each form with n fields shows at the model end as having $n + 1$ fields. The identity field value serves as a marker for each user private space in case they want to keep their data separate from other users.

3.3 Discussion

The system learns over new form instances, each time a user submits an internet form. If the NBC's already exist, they are updated with the probabilities stemming from the current values. If they do not exist, they are created. Note that this technique allows for remote web pages to be modified, without loosing the partial learned models. If a new field is added, no NBC exists for it, so one will be created, as well as leaf nodes in al other NBC's comprising the current strategy for the form. If an existing field is removed, then it will never appear in the evidence, or in the new data coming from that form, and it will be safely ignored both in the updates to the model and in the calculation of probabilities.

We say that the model is **distinguishing** a certain value, if it's current probability is above a certain threshold. For the practical uses of the system this value has been chosen to be 0.75. By choosing this threshold above 0.5 it is assured that at most there will be one value the model is distinguishing.

Consider the following example: Mary forms part of a community using the form-filling system. She is a newly hired employee for company ABC and is going to travel from her hometown of San Francisco to Boston for a conference. She is required to fill a travel authorization form in the company's intranet before she can book her ticket.

Since many other users in her community have already used that intranet form, there is a strategy available for that particular form. Mary starts at a field asking for her name. There are many possible names available (all the names of people who have filled the form before her). In general, the prior probability distribution over those possible values should be fairly uniform. This is unless there are people in the community who travel much more than others. In that case, the highest probability value is unlikely to be distinguishing, considering that there is also the *identity* field that is going to lower all probabilities across the board. Therefore, the system can not offer an automatic suggestion to Mary.

Later in filling the travel authorization form Mary focuses on a field asking for the origin city. Since most of her coworkers leave from San Francisco airport, this city is automatically proposed to her. When Mary arrives to the destination field, none is automatically offered, just a list that shows that most people in her company travel to Boston and New York, but with probabilities nowhere near a distinguishing value. She select Boston from the offered list, and moves on to a field asking a destination state. Based on the current evidence (which include Mary's name, and origin and destination cities) it is plausible to see that the NBC for destination state will choose Massachusetts: the name offers no interesting correlation, and neither does the origin city.

The previous example offers a user centric view of how a system using the described strategy would be helped. The UUID identity field is meant to bias otherwise proportionally equivalent probable values towards values associated with the current user. Note that this bias does not mean that only data associated with the user will be offered. By being bayesian about how evidence reinforces beliefs, this bias can be overcome if a higher amount of data is being offered as evidence towards a contradicting value.

Finally, the example shows how the data from the community of users is useful in cases where *non-personal* data is being entered (i.e correlation between Boston and Massachusetts). If the learned models are not transferrable across forms, that means that a user is doomed to re-enter all her personal information whenever a new form is encountered. This is the main reason to build on the present structure to provide a schema alignment between forms that will allow for a certain amount of data transfer between the models.

4 Online Form Schema Matching

The procedure described in the previous section builds a probabilistic model for each form that is being accessed by the community of users using the form-filling system. Since the models are not shared between forms, it is natural to extend the system to achieve a certain amount of model sharing. The most basic problem is that a zip code form field in a newspaper registration form is basically the same as the zip code we enter when buying online.

As discussed there is a distinct separation of schema matching approaches. One uses the schema (metadata) itself and other uses the instances (data) of the schema (this does not preclude hybrid approaches). Using the schemas is specially difficult on form fields, since the schemas are only present as field names. Many websites are engineered through application servers and tiered approaches that render the field names very opaque. Therefore, the approach used is instance and probability based.

4.1 Definitions

Given two forms present in the form database, $F = f_1, f2, ..., f_n$ and $G = g_1, g_2, ..., g_m$, a schema alignment $\mathcal{A}$ is a function that assigns a probability to each element in the cross product of the field sets:

$$\mathcal{A}(f_i, g_j) = \begin{cases} 0 & : \quad V(f_i) \bigcap V(g_j) = \emptyset \\ P(f_i \rightarrow g_j) & : \quad otherwise \end{cases} \tag{7}$$

The concept is not symmetrical ($P(f_i \rightarrow g_j) = P(g_j \rightarrow f_i)$ does not necessarily hold). The concept that one field conveys might be subsumed (or subsume) other concepts.

Only fields that have at least one value in common are considered possible matches. A probability of 0.25 is subjectively assumed for fields that share (at least) one value.

Whenever one is using probabilities there is strong advise against assigning zero valued probabilities to events that might have a certain plausibility, however low that might be. This is the case with fields with no intersection, but in this case the space to explore (the cross-product of all possible fields in the cross-product of all learned forms) is so big that it is a sensible approach.

Note that several approaches make the case for handling misspelled values (through string edit distance for example) and the definition above does not rule that option. Misspelling being the exception, all it is required of a column pair is that there is a couple of non misspelled values that are identical. Implementation wise this brings another bonus: the use of efficient relational database indexes to bootstrap the space of possible matches.

4.2 Learning Mappings

In general, $P(f_i \rightarrow g_j)$ is actually not correct, what we really want to evaluate is $P(f_i \rightarrow g_j|\mathcal{D})$: the probability of f_i mapping to g_j given the historical data $\mathcal{D}$). In order to estimate this probability, we use the values belonging to the field f_i we are trying to map and take the values of g_j as a given. Therefore $\mathcal{D}$) really means $V(f_i)$ in this case. As it is known from Bayes rule:

$$P(f_i \rightarrow g_j|V(f_i)) = \frac{P(V(f_i)|f_i \rightarrow g_j)P(f_i \rightarrow g_j)}{P(V(f_i))} \tag{8}$$

Where $P(f_i \rightarrow g_j)$ is the prior probability that f_i maps into g_j, $P(V(f_i))$ is the prior probability of the values themselves and $P(V(f_i)|f_i \rightarrow g_j)$ is the probability of observing the values given that the mapping is true (therefore the values ought to occur in the mapped field).

The alignment in the case of disparate forms is more subtle that in the general case of two schemas concerning similar domains. One user might visit sites A and B, while another might B and C. Data obtained from the second user while evaluating $\mathcal{A}(a_i, b_j)$ will be useless and failure to find such data in A is not indicative of a poor match, it just reflects the fact that the second user does not visit A. Therefore only values that are sufficiently similar (through string similarity functions such as edit distance) to other form field's values are considered.

Bayes rule is used online, as data arrives to estimate the probability of the match. If $P(f_i \rightarrow g_j)^t$ is the current (as of time t) estimation of the probability of f_i mapping into g_j, when a new value v is presented to field f_i, then:

$$\begin{aligned} P(f_i, g_j)^{t+1} &= P(f_i \rightarrow g_j|v) \\ &= \frac{P(v|f_i \rightarrow g_j)P(f_i \rightarrow g_j)^t}{P(v)} \\ &= \frac{P(v|f_i \rightarrow g_j)}{P(v)} P(f_i \rightarrow g_j)^t \end{aligned}$$

Using this rule, the strength of the mappings reinforces or weakens as new data comes in.

4.3 Discussion

The benefits of a probabilistic approach become apparent when the information in the schema alignment is merged with the models developed in the previous section. For example, when trying to fill form field f_i, there might be more than one possible mapping. By assuming independence, the probability of the mapping can be evaluated in conjunction with the probability of the value predicted.

Note that new mappings can be established when at least one user interacts with two forms on the internet. The mappings are available to all users, therefore another user can rip the benefits even when interacting with a new form.

There is always one default mapping between forms and is the one concerning the artificial identity column. When the hypothetical user Mary from the previous section is either faced with a new form or a field offering no "distinguished" values (i.e. no value stands out), the system can resort to mapped fields. In the first case (a new form) the value of the system presents itself prominently when entering personal information. If Mary already entered her first and last name in any form mapped to the new one she is facing, the values will become available. If Mary is visiting a previously visited form, but no value stands out, then the current "un-distinguished" values can be compared to the values available via the mapping, which will be in any case discounted by the probability of the match.

At a practical level, the algorithm works in two steps, first it identifies potential matches, then it follows those matches online. The process of identifying potential matches is a batch (and potentially costly) process over the NBC models. It is performed at larger intervals (i.e. every night), but then those additional new potential matches are followed during each interval.

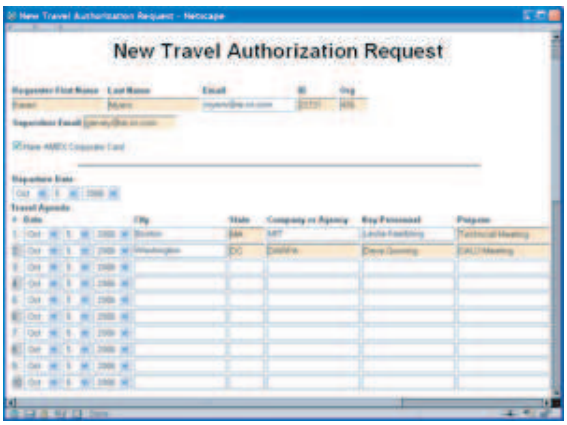

Figure 3. FOAM Plug-in over Intranet Form

5 FOAM

An incarnation of the system described here has been developed at SRI International under the CALO project. It's name is FOAM.

5.1 Software

FOAM is composed of a server and a set of browser plug-ins (one for each browser supported, although development has focused heavily on a Firefox plug-in).

The browser plugin overlays every web page with functionality, in a way similar to to Grease-Monkey scripts. The objectives of the plug-in are:

- Automatically fill the distinguishing values (values with probability above a certain threshold) that the model is identifying
- When there are no distinguishing values, offer an overlayed drop down menu with the n most probable values in decreasing probability order
- Listen to form submissions and report them to FOAM web server so that the models can be updated and new mapping learnt

Figure 1 shows a *Travel Authorization Request* form from an intranet. The shaded form fields have been automatically filled by FOAM, using either the form model or other models connected via the mappings. Each time the user focuses into a new field, the browser will ask the server for the n most probable values and depending on the probability of the first, the field will be automatically filled or not.

From an implementation point of view, the FOAM server generates a UUID that is stored in the browser as a cookie. The browser receives all data from the server via cross domain scripting. This means that FOAM can also be used without a plug-in if it is included in the server.

5.2 Privacy and Security

By default FOAM will ignore *password form fields*, assuming that it is sensitive information and that the benefits of learning over it are out-weighted by the potential security and privacy risk.

Since FOAM is ”listening” to all form submissions, there is an obvious security concern. FOAM will work best if the user leaves it on for most of her browsing day. Even if the information is encrypted between the user browser and the FOAM server, there is a concern that a malicious user could fool the system into believing it is somebody else. Although the identity field makes it harder for such a scheme to work with personal information, it is still possible to enter enough public personal data to overcome that barrier. By using data publicly available about a person, FOAM can start filling in private data that the user does not necessarily want to share.

In order to circumvent this, FOAM can operate in private and public spaces. In the private space only the user data is used to propose values and private models are constructed using only the data provided by the user. In the case the user does not even want to share that information, then the server can collect MD5 hashed values, which can be used to compare signatures against locally stored values.

Note that mappings can be found by applying the algorithm to either the public or the private space. Since each mapping is annotated by a probability, there is a natural order to try them when in need. Mappings found in private spaces are shared with the rest of the community in an anonymous way, preventing secret information about a person to be leaked via an undesired connection.

5.3 Discussion

There are two seemingly orthogonal definitions of the Web 2.0. One relates to the way browsers are able to present more dynamic UIs by using DOM modifications. This trend seems to break the instrumentability of web pages FOAM is exploiting. More and more UI toolkits ignore the browser predefined widgets and paint them themselves and then listen to keyboard strokes. Unless the WHATWG specification is able to push trough some sort of HTML version 5, the approach presented here will not work for long.

The other definition of Web 2.0 relates to cooperative applications that get better as more users make more use of them (for example YouTube or MySpace). In this sense, FOAM is a Web 2.0 application, where the whole can be bigger than the sum of its parts. You can imagine FOAM models reaching more and more of the web as it’s users touch more web pages, and of course as more users start using it. The pure learning approach presents both limitations and advantages. By not using natural language methods to understand what is in the page, the system becomes more flexible. By not using an ontology to guide the discovery of concepts, the emergence of un-planned concepts is allowed.

6 Conclusion

The use of FOAM within the community of SRI has been limited so far, but the results obtained without large amounts of data seems promising. Specially the ”flooding” of repetitive personal information around forms that are somehow touched by members of the community. This is of course where systems like Google Toolbar and RoboForm thrive, but there are reasons to believe that within each company (or each sub-domain) there are new repetitive concepts to be found.

The main contribution of this paper is to provide a tractable way of collecting the mapping of concepts present in the forms of the internet. Even though the author has not found the mapping algorithm anywhere else, it is a fairly straight forward application of bayesian learning. The machine learning algorithm to predict values is also a basic application of Naïve Bayes classifiers. The author believes nevertheless that applied in conjunction, they yield a powerful mechanism to understand

semantic connections. It could be considered a ”sneaky” way to arrive to a partial (form related) promise of the semantic web: even if the web pages the users are visiting are not semantically tagged, FOAM is actually providing the tagging. Forms being one of the main points of interaction of humans with the web, it seems fundamental to be able to understand how the concepts in them (the forms) relate to one another.

In a strict sense, the method shown here is not assigning semantics to the concepts represented by form fields. Nevertheless, it is effectively generating clusters. Since many of the applications of schema matching are geared towards query unification, the method presented goes a long way in that direction. Assigning semantics means labeling those clusters with entities coming from an ontology, or even borrowing an ontology from one or more pages from the semantic web. Assuming the semantic web is a subset of the entire web, the method presented is also a way to make possible the concepts present in the semantic way make their way into the ”un-semantic web”.

6.1 Future Work

The approach presented here works only for categorical information. One obvious avenue to pursue is the expanding the models to handle numerical values. This is well supported by Naïve Bayes classifiers, and would require little tweaking in the case of the bayeasian learner for mappings.

Another interesting avenue of research is exploring the transitivity of the mappings. Connections derived through transitivity can bootstrap the learning or even help assign more sensible probabilities to initial mappings.

7 Bibliography

To be completed.