Form Annotation Approach to Long-Tail Process Automation

Sung Wook Kim[1] Hye-Young Paik[1]

[1] University of New South Wales
Sydney 2052, Australia
{skim, hpaik}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-1112
October 2011

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Simple form of annotations, such as tagging, are proven to be helpful to end-users in organising and managing large amount of resources (e.g., photos, documents). In this paper, we take a first step in applying annotation to forms, one of the main artefacts that make up the long-tail of the processes, to explore potential benefits of helping people with little or no technical background to automate the long-tail of the processes. An analysis of real-world forms was conducted to design algorithms for tag recommendations. Our initial evaluation suggests that useful tag recommendation can be generated based on the contents and the metadata of the forms.

We also briefly present FormSys^{+}, a framework for supporting form-based processes. The architecture supports an end-to-end lifecycle of forms, starting from its creation, annotation, and ultimately to its execution in a process.

1 Introduction

Adding metadata such as keywords or tags, is an effective way of categorising a large pool of resource for both personal and public purposes [5, 6]. The act of "tagging", due to its informality and flexibility in use, has been widely accepted by users for future browsing or searching in web-based systems like Flickr, Delicious, and Youtube [6, 8, 15] . The motivations users have for sharing contents also attributes to the popularity of annotation. These include organisation for general public, communication with friends and family, and personal satisfaction [5]. A large body of research exists in various aspects of annotation such as tag usage [9, 16], tagging motivation [5, 16, 21] as well as tag recommendation strategies [23] and tooling supports for annotation [7, 12, 22, 15] of online contents such as photos and social bookmarks. We leverage some of the lessons learned from the previous research work to design and implement a form annotation framework called FormSys$^+$ which collects domain knowledge from the form owners into the system's Knowledge Base (KB) to help modelling and execution of the form-based processes which are prevalent in every part of our daily life such as applying for a drivers licence, opening a new bank account, or applying for a grant or travel within an organisation. We envision that form annotations can ease many aspects of form-based process modelling (e.g., form selection, discovery, input field mapping, and etc) and execution. However, we recognise the fact that manual tagging can be very time-consuming and tedious, and hence we present a list of tag recommendations as a part of the tooling support to ease the burden on the form owners.

FormSys$^+$ is an extension to our previous work called FormSys [24], in which PDF forms[1] are simply uploaded (like any other file-upload in a browser) to FormSys repository to automatically generate matching Web services. These Web services are then used by BPM designers to design and implement form-based processes. When a form is uploaded by the user, FormSys can dynamically generate two services:

- **soap2pdf**: receives data from an application, fills a form with it, and returns the form via email or an URL where the filled form is available.
- **pdf2soap**: extracts the data from a filled form, assembles and sends a SOAP message to an application.

The creation of Web service is based on WSDL/SOAP standards and the procedure is completely automated and hidden to the end-users. The idea in FormSys is that the end-users can easily create Web services out of forms used in their daily tasks - and some of them can be utilised in automation with the help of BPM professionals. Our current work aims to mature the tool further towards the end-users by supporting a form annotation so that the form owners can get benefits of annotation in managing forms as well as process modelling.

Our contributions focus on supporting the tagging activities for forms. Specifically, we:

- Perform analysis on the forms being used in real-world and use it to generate tag recommendations.

[1] To be precise, we use AcroForm, a sub-standard of PDF which contains editable and interactive PDF form elements.

- Design and implementation of tag recommendation strategy; specifically, for input field and form description annotation.
- Design and implementation of form annotation framework.
- Design and implementation of form-based process execution environment.
- Evaluation of input field tag recommendation strategy.

The remainder of the paper is structured as follows. More in-depth discussion on the form-based processes and why we are interested in automating the form-based processes are presented with an illustrating scenario in Section 2, followed by the analysis of the real-world forms in Section 3. In Section 4 we present tag recommendation strategies for input field and form description annotation which are based on the heuristics derived from the analysis results in Section 3. The prototype implementation and the evaluation is described in Section 5 followed by related work in Section 6. Finally, in Section 7 we come to the conclusions and discuss possible future works.

2 Form-based Processes

A business process is a collection of tasks performed to achieve a business goal within an organisation. Due to the complexity and variability of business processes, organisations use information technology to support their business processes. Business Process Management Systems (BPMS) are widely accepted software systems [25] that enable organisations to automate and continuously improve business processes.

Although the benefits of BPMS have been widely recognised [11, 10] there are still a large portion of business processes that are not adequately addresses by these systems. This is because BPMS is not suitable for processes that heavily depends on manual human workflow and customisations [20]. These types of processes make up the "long-tail of the processes", i.e. highly customised activities that are organisation-specific and often considered not critical to the performance of the organisation.

Forms are the main artefacts of the long-tail processes that enables cost effective and simple way of running and managing business processes. However, the use of the forms requires the end-users like employees or students to do more manual work such as downloading the forms, typing in same information repeatedly, and getting approvals from several people from different organisation units. Moreover, the end-users are often not familiar with the process and this costs them a lot of time and energy just to figure out which forms to fill in. The following real-world scenario outlines some of the drawbacks of using a form-based process.

Illustrating Scenarios: Academics and research students in the School of Computer Science and Engineering (CSE) at UNSW must complete a travel request process for work-related travels (e.g., attending a conference). The procedures and policies are described in a web page[1]. There are up to five forms involved in the process. Depending on the employment status or position held

[1]CSE Travel Page, http://www.cse.unsw.edu.au/people/fipras/travel/

in the school, people need to choose a different set of forms (e.g., students are not required to fill in Teaching Arrangement Form) which are available from and managed by different business organisations in CSE/UNSW.

Despite these drawbacks, forms are still prevalent due to the fact that it requires heavy investment in IT and people in order to automate these processes with BPMS solutions or by implementing customised solutions, and many organisations do not see significant ROI in doing this [14].

Our current work aims to resolve this issue by supporting a **Form Annotation**. - We introduce an annotation step where the form owners annotate the forms with meaningful tags, and use this information to help the form owners to easily model and deploy new form-based process that the end-users can select and execute without a hassle. To help the form owners with annotating the forms, we use various heuristics to generate tag recommendations which will be presented in more detail in Section 4.

Form Annotation: The form services are represented by WSDL, and currently there is no other mechanism to supplement how each form is described. From our own experience with the earlier version of the system, we have learned that there is a great need to enrich the form service description. For example, the text field names obtained from the PDF processing library[2] are not clean or complete to generate meaningful input field names in the corresponding SOAP messages. The rich business knowledge of the form owner about particularises about the usage of the form (e.g., in which process the form is used, who should use it) are not reflected in the automatically generated services.

To accommodate this, we designed a form annotation schema and the form owners are asked to perform various tasks to support the annotation process. Form owners are engaged in two separate levels of annotation activities for a form: input fields and form directives.

- **Input Fields**: the form owner chooses descriptive, human-readable names for the form fields using tags. This information is used to provide possible data matching between different forms for input data sharing.
- **Form Directives**: the form owner chooses descriptive/representative tags for the form, she also annotates the form with conditions and rules that may apply to the usage of the form (e.g., approver, other forms that are used together). This information is used in form discovery/selection, and process execution.

In this paper, we focus on input field and form description annotation which are used by FormSys$^+$ for input field mapping recommendations and form discoveries respectively.

3 Analysis of Forms and Annotation Knowledge Base

For generating a tag recommendation list, we use the actual text appearing in the forms, as they are the most likely terms the form owner is going to

[2] iText, www.itextpdf.com

Table 3.1: Different Categories of Input Fields: *Personal Details* vs. *Process Specific*

	Distinct	∼Reused	Reused	#Reused	Average
Personal Details	98	42	56	184	1.88
Process Specific	909	870	39	101	0.11

∼: Not, #: Number of times

Table 3.2: Types of Reused Personal Details Input Fields

	Forms	Reused	Textfield	Checkbox
UNSW FIPRAS	6	8	8	0
NSW Licence	3	11	9	2
OSU HR	5	6	6	0
Ontario	3	14	14	0
BNZ	7	9	9	0
ARC	7	8	8	0

find relevant for annotations. To understand the text extraction issue better, we studied 30 publicly available forms from 6 different organisations: UNSW FIPRAS[1], New South Wales Government Licence, Ohio State University Human Resources, Government of Ontario, Bank of New Zealand, and Australian Research Council.

Input Field Annotation: We first observe that the input fields in forms can be categorised as follows:

- *Personal Details*: Input fields that require personal information such as a name, address, or phone number of the form user.
- *Process Specific*: Input fields that are relevant to the business process the form is used for. For example, in a CSE travel form, this category includes fields such as a location, name, or a date of the conference.
- *Approval Related*: Input fields reserved for approval steps of the process. This includes the name of the approver and the date of approval.

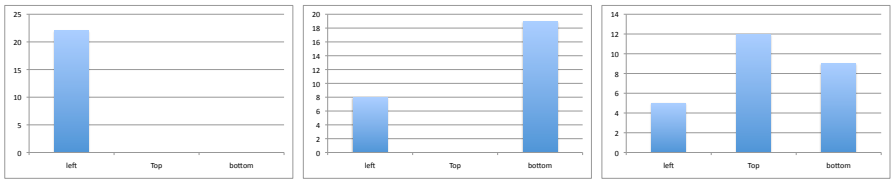
For the input field annotation, we are interested in the first two categories.

Personal Details vs. *Process Specific* – The analysis showed (Table 3.1) that 56 out of 98 distinct input fields in the *Personal Details* category appear in other forms 184 times, which means that each input field were reused 1.88 times on average, compared to 0.11 times for the the input fields in the *Process Specific* category. This tells us that personal detail fields are much more likely to be reused when there are more than one form involved in a single process. Therefore the heuristics for generating a tag recommendation list target input fields in the *Personal Details* category in order to maximise the benefit.
Different Input Types – There are various types of input fields in forms including a textfield, checkbox, and radio button. Table 3.2 shows a number of times

[1]The Finance, Procurement and Assets Unit within School of Computer Science and Engineering

a textfield and a checkbox were reused for personal details input fields. It shows that majority of the input field being reused are of type textfield. This indicates that we can narrow down the extraction candidates to textfields related to personal details.
Positioning Text Labels – Next, we investigated where the relevant texts can be found within the forms for the input fields we are interested in. For each form, we analysed the positioning of the text we want to extract relative to each textfield used to for personal details.

The result (Figure 3.1) shows that some organisations tend to be consistent in the label position they use. For example, reused personal input fields in UNSW FIPRAS forms always had text labels on the left(3.1(a)), and for the OSU HR, majority of the input fields had a text label at the bottom (3.1(b)). Of course, it is not always true for every organisations. For example, the text labels for NSW Licence department forms(3.1(c)) were evenly placed on top, bottom, and left of each input field.



(a) UNSW FIPRAS (b) OSU HR (c) NSW Licence

Figure 3.1: The position of texts relative to the text fields of personal details category

Form Directives Annotation: The form directive annotation we focus on for this paper is form description tag. There are two sources of information used to generate the list of tag form description tag recommendation which are: i) the words extracted from the form and the number of times the word appears in the document, and ii) the meta-data associated with the forms which contains information about the title and subject of the form.

Annotation Knowledge Base: Based on the observations, we first extract vocabularies that will form our initial "tag library" for input field annotation which are the labels related to personal details fields. This forms a part of KB that FormSys$^+$ uses to select a list of input field tag recommendation. When an input field is annotated with a new tag that did not exist previously in the tag library, it is added to the library so that they are available for tag recommendation for future forms.

4 Recommending Annotation Tags

4.1 Input Field Annotation

Once we have the text extracted for each input fields, we need to select a list of tags from the tag library to generate a tag recommendation list. The function

findTagRecommendation listed in Table 4.1 returns a tag recommendation list for a given input field. The list is ranked using MS (Match Score).

Table 4.1: Overview of function findTagRecommendation

FUNCTION	findTagRecommendation
INPUTS	$t_i \in T$, $wo_i \in W$ where, T is the set of all tags in the tag library, $T = \{t_1, t_2, t_3, \ldots, t_n\}$ and W is the set of words extracted from the form by the text extractor denoted by $W = \{wo_1, wo_2, wo_3, \ldots, wo_m\}$
OUTPUT	$R_i = \{(t_i, MS_i), (t_j, MS_j), (t_k, MS_k), \ldots, (t_l, MS_l)\}$ where, R_i is the list of tag recommendation for input field i and MS_i is the Match Score calculated for the tag t_i ($MS_i \in [0,1]$)

The MS is calculated based on three different measures: Label Match (*LabelMatch*), Position Match(*PosMatch*), and Id Match(*IdMatch*). *LabelMatch* provides linguistic similarity between the extracted texts and the tags in the tag library, *PosMatch* takes into account the position of the extracted texts relative to the input field, and *IdMatch* provides linguistic similarity between the PDF Acroform ID of the input field and the tags in the tag library. The MS is calculated as the weighted average of these three measures as shown in Equation 4.1.

$$MS_i = \frac{w1 * LabelMatch_i + w2 * PosMatch_i + w3 * IdMatch_i}{w1 + w2 + w3}$$
$$\text{where,} \quad (0 \leq w1 \leq 1) \quad (0 \leq w2 \leq 1) \quad (0 \leq w3 \leq 1) \quad (PosMatch \in [0,1]) \tag{4.1}$$

Weight w1, w2, w3 indicates the contribution of *LabelMatch*, *PosMatch*, and *IdMatch* respectively. The field id does not always have a meaningful name, but if the creator of the PDF form has assigned a meaningful id to the fields (e.g. words describing the field instead of randomly generated alphanumerics), it is a very useful source of information we can use. Therefore w3 is assigned twice as much weight than w1 and w2 by default.

***LabelMatch*:** The extracted texts are first tokenised based on the new line character. Then it removes unnecessary words from the list of string tokens, using a stop-word list. After the extracted texts are tailored, FormSys$^+$ enumerates all tags in the tag library and matches string tokens with the tags using *Ngram* algorithm. The *Ngram* algorithm calculates the similarity by dividing number of common N-grams by the total number of N-grams.

***PosMatch*:** We take an advantage of the fact that some business units tends to put the text label for personal information input fields in a consistent position for the forms they manage (as shown in Figure 3.1). This predominant position each business units use to position the text label of the input field is referred to

Table 4.2: Weight values for calculating MS

Condition	w1	w2	w3
Default	0.25	0.25	0.5
Label Match == 1, PosMatch == 1	1	0	0
Label Match == 1, IdMatch == 1	1	0	0
Label Match == 1, PosMatch == 0, 0 < IdMatch < 1	0.5	0	0.5
Label Match == 0, PosMatch == 0, 0 < IdMatch	0	0	1
Label Match == 0, IdMatch == 0	0	0	0

Table 4.3: Overview of function findFormDescriptionTags

FUNCTION	findFormDescriptionTags
INPUTS	$wo_i \in W$ where, W is the set of words extracted from the form by the text extractor denoted by $W = \{wo_0, wo_1, wo_2, \ldots, wo_m\}$
OUTPUT	$R = \{(wo_i, MS_i), (wo_j, MS_j), (wo_k, MS_k), \ldots, (wo_l, MS_l)\}$ where, R is the tag recommendation list for form descriptions and MS_i is the Match Score calculated for the word wo_i

as a *regular* position for a given organisation. For example, the UNSW FIPRAS tends to put the text label on the left side of the textfield(Figure 3.1(a)), therefore its *regular* position would be *left*. The term *irregular* position is used to denote the text labels that are not positioned in the *regular* position. For example, input fields that have a text label on the left of the textfield in case of OSU HR(Figure 3.1(b)) are referred to as positioned in a *irregular* position. We assume that KB has information about where the *regular* position is for each business units so that we determine the *regular* position of each form by checking its form owner's business unit.

$$\text{PosMatch}_i = \begin{cases} 1 & \text{if (pos} \in \text{regular} \wedge\ 0\ < \text{labelmatch}_i) \\ 0 & \text{if pos} \notin \text{regular} \end{cases}$$
$$\text{where, pos} = \text{position of text extracted}$$
$$\text{regular} \in [\text{left, top, bottom}] \tag{4.2}$$

The Equation 4.2 shows that for a given input field, *PosMatch* function returns a score of 1 if the tag matches a text extracted from a *regular* position, and returns 0 otherwise.

***IdMatch*:** The input field Id generated during the PDF form creation time is used to provide extra information for tag recommendation since we expect that many form developers will use reasonable Id that is proximate to the text label. *IdMatch* function also takes same approach as *LabelMatch* function. The mapper enumerates each textfield in the form and applies these rules to select tag recommendation from the tag library.

4.2 Form Directives Annotation

The function *findFormDescriptionTags* listed in Table 4.3 returns the tag recommendation list for form descriptions for a given form. It extracts words from the form, and then excludes unnecessary words using a stop-word list. After the extracted texts are tailored, FormSys$^+$ enumerates each word in the list and calculates the MS (Match Score) for each word.

The MS for form description tag is calculated based on 3 different measures which are Subject Match(*SubMatch*), Title Match(*TitleMatch*), and Count Score(*CountScore*). The *SubMatch* and *TitleMatch* matches the word set W with the subject and title metadata of the form. A *SubMatch* score of 1 is given to the word that is found in the subject metadata, and a *TitleMatch* score of 1 is given to the word that is found in the title metadata. The *CountScore* is calculated based on the number of times a word is found in the form. The MS is calculated

as the weighted average of these 3 measures as shown in Equation 4.3.

$$\mathrm{MS}_i = \frac{\mathrm{w1} * \mathrm{SubMatch}_i + \mathrm{w2} * \mathrm{TitleMatch}_i + \mathrm{w3} * \frac{\mathrm{CountScore}_i}{\mathrm{MaxCountScore}}}{\mathrm{w1} + \mathrm{w2} + \mathrm{w3}}$$
$$\text{where,} \quad \mathrm{w1} = 0.4,\ \mathrm{w2} = 0.4,\ \mathrm{w3} = 0.2$$
$$(\mathrm{SubMatch},\ \mathrm{TitleMatch} \in [0,1]) \quad (4.3)$$

Weight w1, w2, w3 indicates the contribution of *SubMatch*, *TitleMatch*, and *CountScore* respectively. The subject and title metadata is not always present, but if the form developer has assigned a meaningful metadata to the fields, it is most significant source of information we can use. Therefore w1 and w2 is assigned a bigger weight than w3 by default. The *CountScore* for each word is divided by the maximum count score(*MaxCountScore*) obtained out of all available words in order to normalise a score between 0 to 1.

5 Implementation and Evaluation

The architecture we adopted is shown in Figure 5.1. The forms uploaded to the Form Repository(1) are subject to annotations before they are deployed. FormSys Core Component(2) is responsible for the retrieval of forms and forms' metadata such as the position of the input fields for the Text Extractor(3) to extract the basis information. The tags that are available in Tag Library(6) are used by the Matcher(4) to generate the Tag Recommendation list(5). At form uploading and process modelling phase, the FormSys Web Front-End(7) provides an interface between the form owners and the system so that the forms are annotated and deployed as a service. At process execution phase, the FormSys Web Front-End interact with the Execution Engine(8) which relies on the process model to determine the flow of execution, and fills in the forms with end-users input data using soap2pdf(9) APIs. Upon the completion of the process execution, the filled-in forms are either emailed to a designated person for an approval or the URL is presented to the form user for downloading.

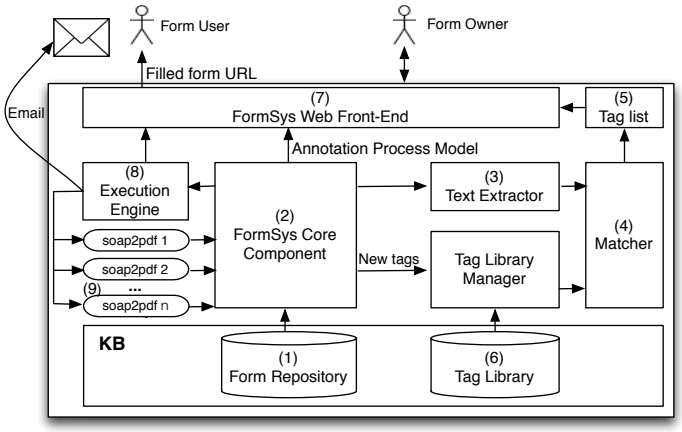
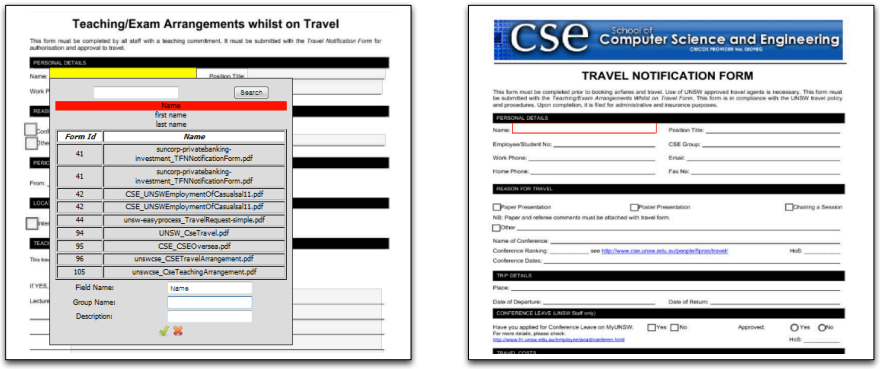


Figure 5.1: FormSys$^+$- Architecture



(a) Tag recommendation for an input field (b) Same tag in another form

Figure 5.2: Input field annotation performed by the form owner

5.1 Case Study Implementation: CSE Travel Request Process

To demonstrate the applicability of our approaches to a real-world case, we took an example of form-based process from our own school which was described in Section 2. It can be demonstrated in two different perspectives. One from the perspective of the form owners who upload the forms and deploy processes, and from the perspective of the form user, in this case, to request a travel.

Figure 5.2(a) shows a tag recommendation list that is automatically generated by FormSys$^+$ for a input field *name* in the Teaching Arrangement form. In this case, the FormSys$^+$ came up with three tag recommendations which are *name*, *first name*, and *last name*. When a *name* tag is selected, it also show a list of forms that contain a *name* tag. When the name of these forms is clicked, FormSys$^+$ opens the form image and shows the actual input field that is annotated with this tag as shown in 5.2(b). This gives a visibility of how the tags are being used within the system, hence helps the form owners to decide which tag name to use for a particular process.

Figure 5.3 shows the process executed by the form user. When a process is deployed successfully, FormSys$^+$ provides a link that the form user can click to initiate the process. FormSys$^+$ provides an execution environment as if the end-user is filling in the actual form document by displaying the form image in the background and overlaying textfields in appropriate positions for the form user to input data. FormSys$^+$ displays a single page at a time and provides a left and right button(2) to flip though the pages. When the first form of the process is completed, the form user can move on to the next form document by clicking next form button(3) or go back to the previous form by clicking the previous form button(4). When the form user moves to the next form, the input data entered in the previous form(5) is passed on to the next form and is used to fill in the input field with same tag name(6). Once all forms are completed (submit button(7)), the execution engine invokes each form service to fill in the form. Once the process is completed successfully, a link(8) to the filled in form documents is returned to the form user.

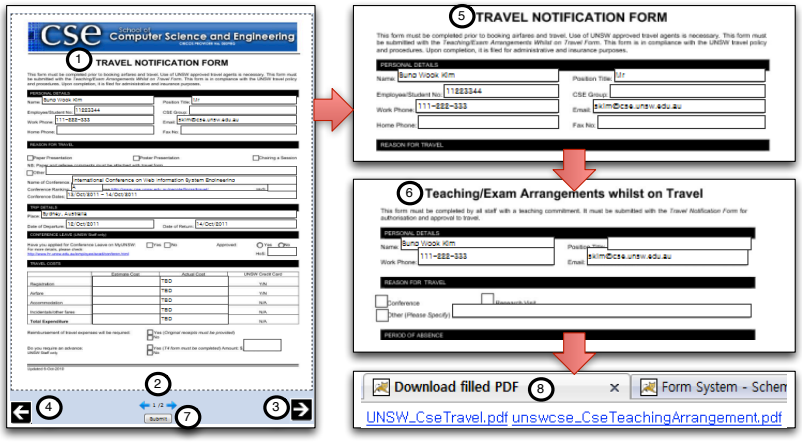


Figure 5.3: Process Executed by the End-users

Table 5.1: Evaluation results for our tag recommendation strategy for personal details input fields

Organisation	Forms	Input fields	S@1	S@5	AMS
UNSW FIPRAS	6	31	90%	97%	0.81
NSW Licence	3	30	47%	73%	0.41
OSU HR	5	33	61%	84%	0.65
Ontario	3	42	64%	86%	0.66
BNZ	7	49	88%	98%	0.80
ARC	7	37	100%	100%	0.99
Average			76%	90%	0.68

5.2 Evaluation of Text Extraction from Forms

For the evaluation of our text extraction approach, we focus on evaluating how well our text extraction and matching score works in selecting a proper tag from the tag library. We make an assumption that we have a populated tag library that contains input field labels for personal information for every form we test, and use common sense to decide whether we have a proper tag recommendations or not. We adopted two metrics, that captures the accuracy of tag recommendation at different aspects:

Success at rank k (S@k) represents the probability of finding a good descriptive tag among the top k recommended tags. For this evaluation, we used two values of k: S@1 and S@5.

Average Match Score(AMS) represents the match score of first relevant tag returned by the system, averaged over all input fields that found a relevant tag. An AMS value is calculated per organisation.

Based on the evaluation result shown in Table 5.1, we observed that 76% of the time, the proper tag for each personal details input field received the highest score amongst a list of recommended tags, and 90% of the time, the proper tag was found amongst top 5 tag recommendations. The Average Match Score for input fields in 30 forms from 6 different organisations were 0.68. In observing

input fields that did not retrieve a proper tag, we have found three main reasons why FormSys$^+$ were not able to generate a proper tag recommendation. First of all, it was due to *irregular* position of the text label. When it's difficult to determine which is the *regular* position for a certain organisation (i.e., positions of the text labels are not consistent), it is difficult for FormSys$^+$ to decide which tag is more relevant than the other when there are raw texts extracted from more than one position. Second, some input fields have rather descriptive text labels (e.g., Address (*licence address must be within NSW*)), and since input field tag recommendation strategy relies on the *Ngram* algorithm to compare the strings, more descriptive text label tends to hamper finding a proper tag recommendation. The third reason is somewhat relevant to the previous point. It is the size of the rectangle we use to extract raw texts. Since the length of the text labels is all different, the size of the rectangle could be too small that it did not extract enough information to determine a proper tag, or it could be too big that the size of the raw texts extracted is too big for accurate recommendation.

6 Related Work

Annotation of resources and its tooling support has been an active field of research. A recent study by Ames et al. [5] on the motivation for annotation in Flickr and the role of tag suggestion in the system has provided number of implications for the design of annotation system in general such as making the annotation pervasive and multi-functional, and not forcing the users to annotate. Their work also reveals that easy annotation features and relevant tag suggestions encouraged the tagging and gave users direction as to the sort of tags they should use. Marlow et al. [15] provides some insights for design decisions in architecting new tagging systems. They point out the importance of studying the incentives for driving participation, and the level of system support to embrace or limit these motivations. The incentives for annotation of online contents such as photos are a well studied area [5, 16]. [5] claims that the motivation for adding tags for the general public is largely due to gain reputation within a community. This motivation is hardly applicable to form annotators since the social impacts of form annotation is much smaller than sharing photos or bookmarks on websites such as Flickr or Delicious, where millions of users upload and share new contents daily. However, we expect that form annotation can contribute greatly to semi-automating the process modelling in public or private organisations at a low cost, in addition to the form organisation and retrieval benefits.

Incentives for annotation is not limited to media contents. It is also applicable for Web service composition. METEOR-S framework [17, 18] proposes a Web service annotation system where an existing ontology (e.g., expressed in RDF) is matched with WSDL documents to generate semantic markups for WSDL elements. However, many research work in this category of annotation assumes an existing (formal) ontology which may not be feasible in many long-tail process scenarios. [19] seeks to extract domain ontologies for Web services from textual content attached to the services. Another work [13] proposes a large-scale annotation framework that does not rely on existing ontologies, but builds its vocabulary dynamically during the process. These are complementary to our work in that our initial annotation knowledge base construction follows

similar approaches and these work can help during the initial bootstrapping phase of the FormSys$^+$ lifecycle.

We see the commercial work in this area as an important and relevant work we can learn from and contribute to. Microsoft SharePoint [1] suite provides a rapid development environment for, among other things, custom design forms and workflows. Its tool for managing forms allows the users to create forms and apply conditional formatting, actions and validations. Its workflow design tool is able to support creation and execution of sophisticated approval workflows. Liferay suite (e.g., Social Office and Portal [2]) focuses on the synchronisation of the work and communication between the team members. However, it has limited support for process automation. For example, its workflow engine Kaleo is used to control creation, approval and rejection by a person of one "asset" (e.g., a document), whereas a form-based process may involves multiple of such "asset". There are many other tools (e.g., Adobe LiveCycle ES2 [3], SAP Gravity [4]) in this category we can discuss, but the strong difference is that our focus is in supporting form-based processes in its "AS-IS" state (i.e., low cost, time and effort), which means we do not require the forms or user interfaces to be designed and created from scratch. We use the forms that are in situ and the users will always see them as the main interacting components for the processes.

7 Conclusions

FormSys$^+$ aims to improve the usability of form-based processes for both form owners and form users without heavy investment in IT infrastructure or human resource. In order to remove the complexity involved in the modelling of business processes, we proposed a new approach using form annotation. However, manual annotation is time-consuming and tedious. Therefore, in this paper, we first presented the results of the real-world form analysis which forms the foundation for the tag recommendation strategy presented in the second part of the paper. We then presented a prototype implementation and evaluation, which showed that the heuristics and algorithms incorporated in FormSys$^+$ works well in generating a tag recommendation for input fields, and lead us to believe that our approach can significantly contribute to the automation of form-based processes. We also presented the execution environment of FormSys$^+$ using an example of CSE travel request process which reuses the input data across multiple forms in a same process, and provides "AS-IS" look-and-feel environment to end-users by using a form as an main interaction object in the user-interface.

Our main idea for FormSys$^+$ is that, it will become an environment where the knowledge contributions from the users are reflected in the continual improvement of the system to support the long-tail processes. Hence, as for future work, we first plan to expand current KB by supporting additional form directives such as conditions or rules that may apply to the usage of the form, and email templates to support simplified email approval request. Secondly, we are investigating different types of matching relationship between the tags to improve data matching between the forms. And we are also looking for ways to utilise the annotations to semi-automate the form discovery/selection processes, and to find possible data matchings between the forms.

Bibliography

[1] Microsoft SharePoint, `http://sharepoint.microsoft.com`, accessed May 2011.

[2] Liferay Social Office and Portal, `http://www.liferay.com`, accessed May 2011.

[3] Adobe LiveCycle EC2, `www.adobe.com/products/livecycle`, accessed May 2011.

[4] SAP Gravity, `http://tiny.cc/gllur`, accessed May 2011.

[5] Morgan Ames and Mor Naaman. Why We Tag: Motivations for Annotation in Mobile and Online Media. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, page 971, San Jose, California, USA, 2007.

[6] C. Cattuto, V. Loreto, and L. Pietronero. Semiotic dynamics and collaborative tagging. *Proceedings of the National Academy of Sciences*, 104(5):1461–1464, jan 2007.

[7] Andreas Girgensohn, John Adcock, Matthew Cooper, Jonathan Foote, and Lynn Wilcox. Simplifying the management of large photo collections. In *Proceedings of INTERACT'03, IOS*, pages 196–203. Press, 2003.

[8] S. A. Golder. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, apr 2006.

[9] Scott. Golder and Bernardo A. Huberman. The structure of collaborative tagging systems. Technical report, HP Labs, 2005.

[10] Paul Harmon. CCLA(Charities, Churches, and Local Authorities) Case Study. BPTrends, March 2010. `http://tiny.cc/ov48h`, accessed May 2011.

[11] Paul Harmon. Danish Municipalities Case Study. BPTrends, April 2010. `http://tiny.cc/ov48h`, accessed May 2011.

[12] Allan Kuchinsky, Celine Pering, Michael L. Creech, Dennis Freeze, Bill Serra, and Jacek Gwizdka. FotoFile: A Consumer Multimedia Organization and Retrieval System. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '99*, pages 496–503, Pittsburgh, Pennsylvania, United States, 1999.

[13] Peep Kungas and Marlon Dumas. Cost-Effective semantic annotation of XML schemas and web service interfaces. In *2009 IEEE International Conference on Services Computing*, pages 372–379, Bangalore, India, 2009.

[14] Philip Larson. BPM SUITES AND THE LONG TAIL OF PROCESS AUTOMATION, August 2005. `www.philiplarson.com/docs/bpm-longtail.pdf`, accessed May 2011.

[15] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. HT06, tagging paper, taxonomy, flickr, academic article, to read. In *Proceedings of the seventeenth conference on Hypertext and hypermedia - HYPERTEXT '06*, page 31, Odense, Denmark, 2006.

[16] Oded Nov and Chen Ye. Why do people tag? motivations for photo tagging. *Communications of the ACM*, 53(7):128, jul 2010.

[17] Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, and Kunal Verma. Meteor-s web service annotation framework. In *Proceedings of the 13th conference on World Wide Web - WWW '04*, page 553, New York, NY, USA, 2004.

[18] Preeda Rajasekaran, John Miller, Kunal Verma, and Amit Sheth. Enhancing web services description and discovery to facilitate composition. In Jorge Cardoso and Amit Sheth, editors, *Semantic Web Services and Web Process Composition*, volume 3387, pages 55–68. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[19] Marta Sabou, Chris Wroe, Carole Goble, and Gilad Mishne. Learning domain ontologies for web service descriptions. In *Proceedings of the 14th international conference on World Wide Web - WWW '05*, page 190, Chiba, Japan, 2005.

[20] Terry Schurter. BPM state of the nation 2009. BPM.com. `http://www.bpm.com/bpm-state-of-the-nation-2009.html`, accessed May 2011.

[21] Shilad Sen, Shyong K. Lam, Al Mamunur Rashid, Dan Cosley, Dan Frankowski, Jeremy Osterhouse, F. Maxwell Harper, and John Riedl. tagging, communities, vocabulary, evolution. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work - CSCW '06*, page 181, Banff, Alberta, Canada, 2006.

[22] B. Shneiderman and H. Kang. Direct annotation: a drag-and-drop strategy for labeling photos. In *IEEE International Conference on Information Visualization*, pages 88–95, London, UK, 2000.

[23] Börkur Sigurbjörnsson and Roelof van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceeding of the 17th international conference on World Wide Web - WWW '08*, page 327, Beijing, China, 2008.

[24] Ingo M. Weber, Hye-young Paik, Boualem Benatallah, Zifei Gong, Liangliang Zheng, and Corren Vorwerk. FormSys: Form-processing Web Services. In *Proceedings of the 19th international conference on World wide web - WWW '10*, page 1313, Raleigh, North Carolina, USA, 2010.

[25] Mathias Weske. *Business process management concepts, languages, architectures*. Springer, New York, 2007.