Context-Aware Form Filling for Web Applications

Melanie Hartmann, Max Mühlhäuser
Telecooperation Group
Technische Universität Darmstadt
Darmstadt, Germany
{melanie,max}@tk.informatik.tu-darmstadt.de

***Abstract*— Context-aware user interfaces facilitate the user interaction by suggesting or prefilling data derived from the user's current context. This raises the problem of determining which context information can be used as input for which interaction element in the user interface. This task is especially challenging as the texts that describe the elements, e.g. their labels, often differ in the terminology used.**

In this paper, we present a novel mapping process for that purpose which (i) combines the advantages of string-based and semantic similarity measures to bridge the vocabulary gap between context and UI element, and which (ii) is able to automatically extend its vocabulary by observing the user's interactions. We show that these two features dramatically increase the quality of the resulting mapping. Unlike previous approaches, the proposed mapping process does not require any training or manually tagged data. Further, it does not only use the label to describe the context and UI elements, but additional texts like their tooltips.

***Keywords*-Intelligent User Interfaces; Similarity Measures; Context-Awareness; Context-Aware User Interfaces**

I. Motivation

Nowadays applications get more and more complex. To facilitate the interaction with an application, we need user interfaces (UIs) that provide proactive assistance, for example by suggesting which values to enter in a form. The importance of these proactive suggestions, especially for mobile usage, is stressed by Rukzio [1]. He found that users are four times faster on a smart phone when they just have to correct prefilled form entries compared to entering the information from scratch. Which data should be suggested thereby strongly depends on the user's context. This context information ranges from physical data (like the user's location) to "virtual" data (like calendar entries or data the user entered). For example, data that is entered by the user is especially useful for tasks where the same information needs to be provided in several forms, e.g. when planing a trip which includes renting a car, booking a flight, etc. UIs that are aware of the user's current context, and use it to provide context-aware suggestions are called context-aware UIs. An example screenshot of such a context-aware UI is shown in Figure 1. It suggests the user which data to enter in the input fields by querying the user's calendar and his current location.

In many cases, the terminology used in the UI differs from the one used to describe the context. For example, the location information in a calendar entry has the label "location" whereas the location information for the UI in Figure 1 is labeled with "to". Thus, we need a mapping process

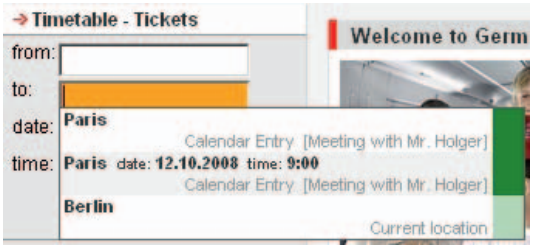


Fig. 1: Example of context-aware interaction support

that is able to bridge the existing vocabulary gap. Thereby, we can neither rely on pre-tagged data, as we cannot tag all possible domains the user might encounter in advance, nor can we collect sufficient training data. Building on our preliminary study in [2], we present a novel approach for mapping context information to UI elements that combines string-based and semantic similarity measures, and also learns from observing the user's interactions.

In the remainder of this paper, we first briefly describe how UI elements and context elements are represented and how context information is gathered. In Section IV, we present our novel mapping process, and describe how the resulting mapping between context and UI elements can be applied by a context-aware UI. Finally, we report on the evaluation of our mapping process.

II. Related work

The task of mapping context information to input elements is strongly related to ontology mapping [3] and database schema matching [4], where concepts have to be mapped to ontology entries or column names, respectively. However, in these areas most approaches benefit from additional information like constraints or instances that are more distinctive than the input element's label alone or they rely on a large amount of training data. Both is not available for context-aware UIs. Also related is the research on the deep web[1], as it is concerned with mapping textual representations of several web forms. For example, Wu et al. [5] use cosine similarity for determining the similarity of label and name attributes. However, it also relies on a large corpus of training data.

[1]Deep web refers to all information in the web that cannot be accessed via conventional search engines following hyperlinks.

978-0-7695-3800-6/09 $26.00 © 2009 IEEE
DOI 10.1109/ICSC.2009.83

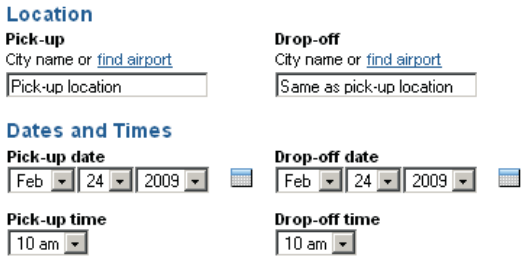


Fig. 2: Example form

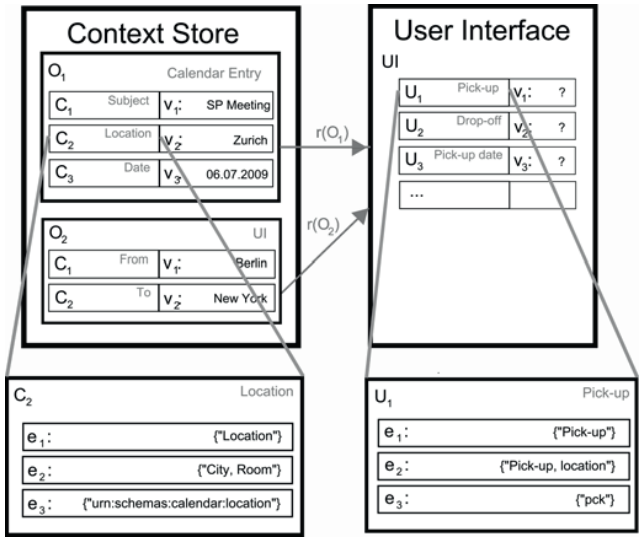


Fig. 3: Example formalization of two context objects and a UI

Other approaches for automatically filling in forms (e.g. [6]) either require apriori tagging of web sites, or a manually crafted list containing the labels or names of input elements which describe a semantic concept. Thus, these approaches can only be applied to a specific domain (in [6] they focus on address information) or need explicit advice by the user.

More generic approaches like [7] use a predefined list of synonyms for the mapping and learn from observation. However, they only consider the elements' labels for that purpose which often does not convey enough information.

Furthermore, in contrast to our approach none of the presented approaches deals with dynamic context information like calendar entries.

III. Representing UI and Context Objects

For mapping context information to UI elements, we first need a representation for UI elements and context information that is stored in the user's *Context Store*. The element's label is thereby often not sufficient for unambiguously describing the element as it sometimes refers to several interaction elements (e.g. "Pick-up date" in Figure 2) or does not convey enough information (e.g. "to" in Figure 1). For that reason, we take all available texts that describe the element into account, in contrast to existing approaches.

In the following, we point out which texts can be used to represent the various elements and how the representation is formalized. A visualization of the formalization is presented in Figure 3.

A. Representing UI elements

When representing UI elements, we focus on HTML web applications as they are widely used. A HTML element provides the following information that can be used to represent the element: (i) its **label** which is human readable (e.g. "Pick-up date" in Figure 2), (ii) the **name** attribute of the input element that gives us its technical label, though this is often not human readable (e.g. "pck" for the "Pick-up" element)[2], (iii) the corresponding **tooltip** ("alt" attribute), (iv) the data that is **prefilled** to give the user a hint (e.g. "Pick-up location" is prefilled in the "Pick-up" elemen in Figure 2), and (v) the **values** in drop down menus, radio buttons, or grouped checkboxes. *Name*, *tooltip*, *prefilled* and *values* can be extracted directly from the HTML representation. The HTML syntax also defines a tag LABEL for marking a *label* of an input element; however, it is scarcely used in practice (only about 20% of the input elements we used for the evaluation had an associated label attribute). For that reason, we extract the label using LabelFinder [2] that analyses the visual layout.

Each UI element U_i can thus be represented as a list of descriptive texts e_r (see also Figure 3). These texts are ordered by their relevance r. This relevance is required as we assume that the best mapping can be determined if we consider only the most relevant information. Further information should only be incorporated if this text does not convey enough information. In initial experiments (see [2]), we determined the following order of relevance according to the expressiveness of the different attributes: *label*, *values*, *tooltip*, *prefilled*, *name*. The individual texts e_r are separated into a set of single words, so-called **tokens**. For example, the first element in the example form in Figure 2 is represented by its label, prefilled and name attributes U_1=(e_1, e_2, e_3)=$(\{Pick\text{-}up\}, \{Pick\text{-}up, location\}, \{pck\})$.

The entire UI is finally represented as a set of elements $\{ U_1, U_2, ...\}$ with corresponding values $\{v_1, v_2, ...\}$. However, these v_i are usually not given in the UI as this is the data that needs to be entered.

B. Representing context objects

The *Context Store* contains all **context objects** O_j that might be relevant for the user's current interaction. This data is derived (i) from an associated context server or (ii) from observing the user's interactions. Each O_j consists of a set of **context elements** C_i with associated values v_i, e.g. a calendar entry like in Figure 3 consists of a subject, a location, and a date element. The goal of context-aware UIs is to determine which context object O_j from the *Context Store* should be

[2] This attribute is often a camel case word consisting of several concatenated words like "arrivalDate" because the attribute may not contain white spaces.

suggested to the user and which of its values v_i should be suggested as input for which UI element U_i.

The elements C_i are represented in analogy to UI elements. If the context element C_i was derived from the user's interaction it is represented by the same attributes as the corresponding UI element. If it was gathered from the context server, we have to rely on the information provided by the context server. This information is at least a label like "start date" which we therefore take as most descriptive text e_1. Sometimes additional information like a brief description is provided by the context server which is represented as further descriptive texts.

IV. Mapping textual representations

Having determined the representation of the available input elements of a UI, we try to find relevant objects in the user's *Context Store* that can serve as input for the UI elements, i.e. which v_i should be suggested for which input element. As we want to be able to provide support for arbitrary domains and UIs, we cannot rely on UI elements that are already tagged with their semantic concept (e.g. from a common ontology) nor on a large amount of training data to learn these semantic concepts. We can only make use of the available textual descriptions as described in the previous section to determine the mapping.

As basis for the mapping process, we first need a similarity measure that computes the similarity between two tokens, e.g. between "Pick-up" and "Location" (see Section IV-A). This similarity measure is then applied to determine the similarity of two elements described by various descriptive texts (see Section IV-B), e.g. the elements C_2 and U_1 in Figure 3. The resulting similarity value between elements is then used to determine the best mapping between context and UI elements, e.g. which elements of O_1 should be assigned to which elements in the UI (see Section IV-C). Which context object should be finally suggested to the user is described in Section V.

A. Computing similarity of tokens

As basis for our mapping process, we need a similarity measure $sim_{token}(a, b)$ which computes the similarity between two tokens a and b on a scale from 0 to 1. For sim_{token}, we can use similarity measures that are based on string comparison (**string-based measures**) or that rely on an additional knowledge base like Wikipedia[3], Wiktionary[4] or WordNet [8] (**semantic measures**). In contrast to string-based measures, the semantic measures recognize similarities between strings that use a different terminology (e.g. "destination" and "airport"). However, they do not recognize the similarity of strings that differ in their spelling variants (e.g. "e-mail" and "email"). as they are usually not reflected in the semantic knowledge bases. In contrast, they can be easily identified by string-based measures.

[3] http://www.wikipedia.org
[4] http://www.wiktionary.org

In a preprocessing step, we turn all tokens in lower case representation, pull apart camel case words and skip all tokens that only consist of one character. Further, we lemmatize all tokens, i.e. use their canonical form (e.g. "mouse" instead of "mice" and "run" instead of "ran"). This is necessary as the knowledge bases that are used for the semantic measures only contain lemmas.

The distribution of the actual sim_{token} values vary for the different measures, e.g. for some measures 0.6 is a relatively low value. To ensure that only really similar tokens are considered in the mapping process, we state a threshold θ for sim_{token} for every similarity measure, i.e. if $sim_{token} < \theta$, then $sim_{token} = 0$.

B. Computing similarity of elements

We compute the similarity $sim^k(C_i, U_j)$ between two elements C_i and U_j as the average similarity for every possible pair of tokens by taking the k most relevant descriptive texts into account. The limitation of considered texts is needed, because we first try to determine a mapping with the most relevant descriptive texts ($k = 1$) as we assume that it returns the best results. Further descriptive texts are only considered for those elements that could not be assigned in the first step (see Section IV-C). We define

$$sim^k(C_i, U_j) = \frac{\sum_{a \in X^k_{C_i}, b \in X^k_{U_j}} sim_{token}(a, b)}{max(|X^k_{C_i}|, |X^k_{U_j}|)}$$

with $X^k_{C_i} = \bigcup_{r \leq k} e_r \in C_i$ and $X^k_{U_j} = \bigcup_{r \leq k} e_r \in U_j$. For example, for $sim^2(C_2, U_1)$ for C_2 of the context object O_1 and U_1 of the example in Figure 3, we determine at first $X^2_{C_2} = \{Location, City, Room\}$ and $X^2_{U_1} = \{Pick\text{-}up, Pick\text{-}up, location, pck\}$. If we use a string based similarity measure for sim_{token} that returns 1 if the tokens are the same and otherwise 0, only $sim_{token}(location, location)$ results 1, all other similarity values in 0. Thus, $sim^2(C_2, U_1) = 1/max(3, 4) = 1/4$.

C. Mapping context and UI elements

For determining the mapping M_m between a context object O_m and the UI, we first identify the best assignments of context to UI elements using only the most relevant descriptive text e_1. We compute $sim^1(C_i, U_j)$ for every element $C_i \in O_m$ and every element $U_j \in UI$. This results in a mapping like the one in Figure 4 (a). M_m initially contains all pairs (C_i, U_j) with a similarity value greater than 0. As we assume that each concept is only used once in a context object or in a UI, the mapping has to be unambiguous. For that reason, we need to solve all generated ties, i.e. to resolve all conflicts in the mapping. The **Solve Ties** process iterates over all pairs $(C_i, U_j) \in M_m$ ordered by their similarity value. Each pair that is in conflict with the currently considered pair, is deleted from M_m. If two pairs have the same similarity measure, both are deleted from the mapping. Figure 4 (b) shows the result of Solve Ties for the example mapping in Figure 4 (a).

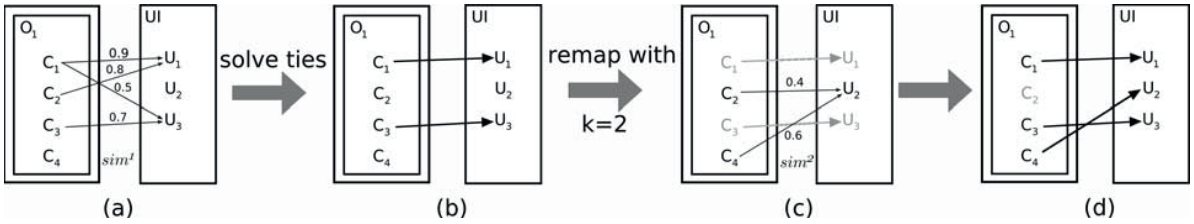


Fig. 4: Applying the mapping process to map a context object to a UI

All elements that are now unambiguously assigned to another element are excluded from the following mapping iteration. We again compute the similarities between the various remaining elements, but this time we take one more descriptive text into account, i.e. we use $sim^2(C_i, U_j)$. Emerging ties are solved and the unambiguously assigned elements also excluded from the next mapping iteration. The process is repeated with further descriptive texts until no more descriptive texts remain or no more assignments could be found. The pseudo-code for the whole process can be found in Algorithm 1.

Which similarity measure sim_{token} is best suited for the mapping process is evaluated in Section VI. As stated before, string-based measures have the advantage that they can be applied to arbitrary domains that use the same terminology, whereas semantic measures can bridge the vocabulary gap between domains, but only if the used knowledge base contains the required terms. To benefit from both types of similarity measures, we extend our process with a **second mapping step**. We at first determine a mapping with one type of measure and then try to map the unassigned elements using the other type of measure. We expect that the string-based mappings provide a better initial mapping than semantic measures, as they usually achieve higher precision values than semantic measures. Thus, they should be applied for the first mapping step and semantic measures for the second.

The existing vocabulary gap between a context and a UI representation cannot always be bridged with the help of semantic similarity measures as the knowledge base used does not contain all relevant terms. For example, the term "pick-up", which is often used for car rental forms, is not contained in any of the knowledge bases that we consider. For that reason, we further apply **learning** in our mapping process. In those cases where the mapping could not be determined, we learn which terms are related to each other from observing the user's input. If we observe that the user has entered data in a UI element U_i that corresponds to a value v_i contained in the user's *Context Store*, we store the label (e_1) of the corresponding C_i as related expression to the label (e_1) of U_i. For example, if the user entered "Zurich" for v_1 in the UI in Figure 3, we store that "Pick-up" (the label of U_1) is related to "Location" (the label of C_2). Note that these expressions can also consist of several tokens (e.g. "Pick-up time"). $sim_{token}(a, b)$ then returns 1 if the token a is contained in an expression that is related to an expression that contains the token b.

Algorithm 1 Mapping Strategy

Require: Set $\{C_i\}$ of elements of a context object O_m
Set $\{U_j\}$ of UI elements
Ensure: A mapping M_m between the elements of O_m and the UI

$CO = \{C_i\}$; $UI = \{U_j\}$; $k = 1$; $M_m = \{\}$;
k_{max} = max number of descriptive texts in CO and UI
while $CO \neq \emptyset \ \wedge UI \neq \emptyset \ \wedge k \leq k_{max}$ **do**
    $M = CO \times UI$
    for all $(C_i, U_j) \in M$ **do**
        Compute $sim^k(C_i, U_j)$
    end for
    Order all elements in M by their similarity
    SOLVE TIES for M
    for all $(C_i, U_j) \in M$ **do**
        Remove C_i from CO
        Remove U_j from UI
    end for
    $M_m = M_m \cup M$
    $k = k + 1$
end while

V. USING THE CONTEXT MAPPING

In order to decide which context object should finally be suggested to the user, we compute the mapping to the UI elements for every context object contained in the *Context Store*. How well a context object O_i fits to the required input is stated by its relevance $r(O_i)$. This relevance $r(O_i)$ is determined by the amount of elements C_j of O_i which could be assigned to UI elements. The more elements could be assigned the more likely it is that O_i is relevant for the current UI. We thus define $r(O_i)$ as

$$r(O_i) = |M_i|/|O_i|$$

i.e. the fraction of elements of O_i that could be assigned to elements of the UI. For example in Figure 4 (d), $r(O_1)$ is $3/4$.

The most relevant context objects are then suggested to the user as visualized in Figure 1. To accept a suggestion the user just has to select the corresponding entry and all associated input fields, radio buttons etc. are filled with the corresponding values. To make the user aware of the certainty of the suggestion, the relevance value $r(O_i)$ is visualized as green box next to the suggestion (ranging rom dark green (100% relevant) to white (0% relevant)).

TABLE I: Micro-average precision, recall and F_1-measure for single domains and across related domains (*: best value)

	Domain	cars			flights			hotels			address			cars&hotels			cars&flights			hotels&flights		
		P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
string-based	exact	.93	.82	.87	.95	.72	.82	.88	.87	.87	.89	.71	.79	.84	.40	.54	.52	.10	.17	.62	.25	**.35***
	b-substr	.94	.83	**.88***	.95	.76	**.84***	.87	.87	.87	.91	.79	**.85***	.79	.39	.52	.50	.12	.19	.54	.24	.34
	jaro	.89	.80	.84	.90	.74	.81	.88	.88	**.88***	.82	.76	.79	.74	.40	.52	.48	.12	.19	.36	.24	.29
	jaro-w	.89	.81	.85	.90	.73	.81	.87	.88	**.88***	.77	.75	.76	.68	.39	.50	.41	.11	.18	.35	.24	.29
	monge-elkan	.94	.83	**.88***	.90	.75	.82	.86	.87	.86	.87	.81	.84	.73	.39	.51	.35	.12	.18	.44	.24	.31
semantic	wordnet	.93	.82	.87	.93	.72	.81	.85	.86	.85	.84	.75	.79	.81	.44	**.57***	.46	.10	.17	.60	.25	**.35***
	wiktionary	.93	.82	.87	.95	.72	.82	.88	.87	.87	.89	.75	.81	.84	.40	.54	.52	.10	.17	.62	.25	**.35***
	wikipedia	.93	.82	.87	.86	.69	.76	.88	.87	.87	.82	.78	.80	.78	.42	.54	.48	.15	**.23***	.47	.25	.33

VI. Evaluation

For evaluating our mapping process, we need a dataset containing possible context objects for a number of web applications and their mappings. As such data is hard to obtain, and we also want to be independent of how the context information is actually represented, we decided to use the representations used in the web forms as possible context representation. This means that we take the representation given by a source web form as a potential context object and try to map it to a target web form. The web forms were taken from several domains, whereby some of them are **related domains**, i.e. the same information can be entered but they often use different terminology. For example, web forms for booking a hotel room and for booking a car are related as they both need a start and end date etc. but use different terminology like "check in" and "pick up". For the evaluation within a **single domain**, we used every combination of source and target web form from this domain. For related domains, we combined one form from a domain with a form from a related domain.

In order to judge the quality of the resulting mapping, we have to consider that often no direct mapping between elements is possible. For example, one representation uses two fields for entering the name, one for the first and one for the last name, while another representation uses only one field for the full name. Assigning the element representing the first name to the one representing the full name is not entirely correct, but it enables us to suggest at least half of the input for the interaction element. For that reason, we include these partially correct assignments in our rating but with a lower weight than correct assignments.[5] For each mapping, we compute the precision and the recall as follows:

$$Precision = \frac{correct + 0.5 \cdot partial}{correct + 0.5 \cdot partial + wrong}$$

$$Recall = \frac{correct + 0.5 \cdot partial}{maxScore}$$

In this formalization, $correct$ denotes the number of correct assignments, $partial$ the number of partially correct assignments and $wrong$ the number of wrong assignments. $maxScore$ is the score that is reached by an optimal mapping.

In the following, we first list the similarity measures that we assessed in our comparison and describe the evaluation data set. Then we report how the different measures perform in a single domain and in related domains[6] (Section VI-A). Next, we show that our proposed mapping process - including a second mapping step using another similarity measure and including learning- increases the overall quality of the mapping (Section VI-B). Finally, we determine how well our approach can distinguish a relevant from an irrelevant context object (Section VI-C).

a) Similarity measures: String-based measures determine the similarity between two strings by comparing their characters. We use two baseline string measures: The exact string match measure (abbreviated as **exact**) returns 1 if the strings are exactly equal, and 0 otherwise. The bounded substring match measure (**b-substr**) returns 1 if the strings have a shared substring of at least 3 characters that is a prefix or a suffix of the other string (this matches strings like "arrival" and "arrive"). We also consider three more sophisticated measures that return a value in the interval $[0, 1]$: (i) the measure by Jaro [9] (abbreviated as **jaro**) that takes typical spelling deviations into account, (ii) an adaptation of the *jaro* measure by Winkler [10] (**jaro-w**) which increases similarity scores in the case of shared prefixes, and (iii) the measure by Monge and Elkan [11] (**monge-elkan**) that uses an affine gap model penalizing many small gaps in the string match more than a large gap. For our experiments, we implemented the *exact* and *b-substr* measures ourselves, and used the SecondString library [12] for the *jaro*, *jaro-w* and *monge-elkan* measures.

Semantic measures use knowledge bases for determining similarities. We use WordNet [8] (abbreviated as **wordnet**), Wikipedia[3] (**wikipedia**), and Wiktionary[4] (**wiktionary**) as knowledge bases for the semantic measures. As considering only the relations contained in WordNet or Wikipedia like synonyms or hyponyms did not yield very good results in our preliminary study [2], we only use concept vector based measures [13] for our evaluation. We used (i) WordNet 3.0 together with the freely available JWNL WordNet API[7], (ii)

[5] This differs from the evaluation in [2], where we filtered all elements that could not produce a correct mapping.

[6] in [2] we only considered single domains

[7] `http://jwordnet.sourceforge.net/`

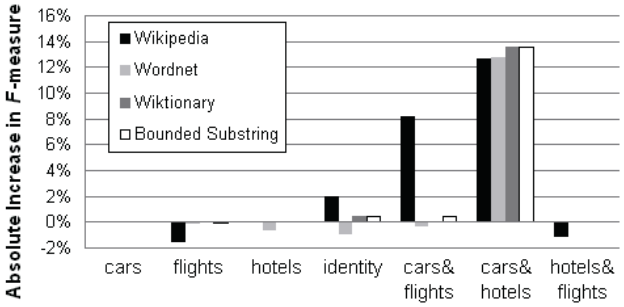


Fig. 5: Absolute difference between the F-measure of a single mapping step using *b-substr* and a two-step mapping process additionally using one of the listed similarity measures for the second mapping.

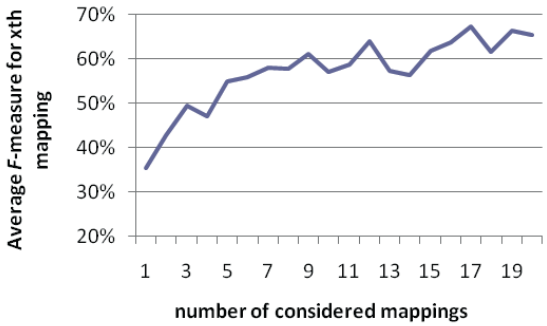


Fig. 6: Average F-measure for the *hotels&flights* domain given the number of observed mappings

the English Wikipedia dump from Feb 6th, 2007 together with the JWPL Wikipedia API[8] [14], and (iii) the English Wiktionary dump from Oct 16th, 2007 with the JWKTL Wiktionary API[8] [14].

For normalizing inflected forms of tokens, we used lemmatization as provided by the TreeTagger [15]. We empirically determined the optimal value of the threshold θ for including similarity values in the mapping process on a dataset that is not used in the experiments. We used the following thresholds: 0.75 for the *jaro*, *jaro-w*, and *monge-elkan* measures and 0.05 for *wordnet*, 0.4 for *wikipedia*, 0.25 for *wiktionary*. All other measures return either 0 or 1, thus no threshold is needed.

b) Evaluation dataset: We took 45 web forms from 4 domains: car rental (consisting of 7 web forms), flights (12), hotels (9), and address (17). Most web forms for the cars, flights, and hotels domains were taken from the TEL-8 dataset of the UIUC dataset [16]. Thus, we have four single domains (*cars*, *flights*, *hotels* and *address*) and three related domains (*cars&flights*, *cars&hotels*, *hotels&flights*).

A. Comparing similarity measures

At first, we compare the baseline performance of the different similarity measures without using learning or a second mapping step. Table I shows the results for the various similarity measures in terms of recall, precision and F-measure[9] for the four single domains and for the three related domains. For the single domains, the results differ slightly on a very high level of performance with an F-measure between .76 and .87. The string-based measures slightly outperform the semantic measures. The performance for the related domains varies heavily, however there are also mostly only minor differences between the various similarity measures. In that setting, the semantic measures perform slightly better than the string-based measures.

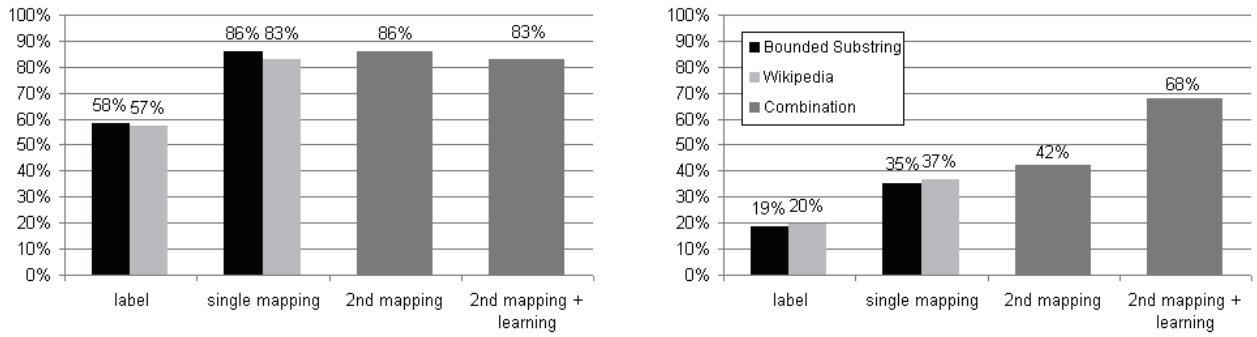
As the *b-substr* measure has the best overall performance of all string-based measures, we consider this measure as representative for all string-based measure in the remainder of the paper. As the semantic measures rely on different knowledge bases, we expect that they vary in their ability to bridge vocabulary gaps. For that reason, we still take all three semantic measures into account.

B. Using 2nd mapping step and learning

In order to improve the quality of the mapping especially across related domains, we evaluated whether a second mapping step with another similarity measure increases the performance. We compared the results for first using a string-based or a semantic measure. As expected, we found that it yields better performance to use a string-based measure first. For that reason, we only report here on the results for using *b-substr* in the first mapping step and a semantic measure in the second mapping step. To evaluate whether a second mapping step itself can increase the performance, we also computed the performance when using *b-substr* for both steps. Figure 5 shows the absolute increase or decrease in the F-measure for the second mapping step compared to using only a single mapping step. A second mapping step dramatically increases the performance by about 13% points for the *cars&hotels* domain no matter which similarity measure is used. This indicates that the second mapping step itself can be of advantage as sometimes some conflicting elements are assigned in the first mapping step so that remaining elements can then be assigned in the second step. Further, *wikipedia* increases the performance for the *cars&flights* domain by 8% points. For the other domains the second mapping step has only marginal influence on the results. This shows that a second mapping step has rarely negative effects on the quality of the mapping, but can increase the performance especially across related domains. As the combination of *b-substr* for the first and *wikipedia* for the second mapping step yields the best results, we only report the results for this combination in the following if not stated otherwise.

Next, we determined the influence of learning related terms from observation on the mapping quality. For the evaluation, we randomly selected 20 pairs of the *hotels&flights* domain and successively computed the mapping while tracking the F-measure for each mapping. This process was repeated 80 times and the results were averaged. Figure 6 shows how the average

[8] `http://www.ukp.tu-darmstadt.de/software/JWPL`
[9] $F = 2 \cdot precision \cdot recall / (precision + recall)$



(a) In a single domain

(b) Across related domains

Fig. 7: Overall performance of our mapping process

F-measure increases with the number of observed mappings. It shows that four observed mappings are already sufficient to increase the F-measure by 40% and ten to reach an overall performance of about 60%.

To be able to estimate the influence of the different factors and to judge the overall performance of the presented mapping process, we compare the F-measures for different configurations for a single domain and across related domains. The result can be seen in Figure 7. As a baseline, we use the standard approach that only considers the label of the input elements (*label*). Using our approach with a single mapping step (*single mapping*) already outperforms *label* by 44% for a single domain and by 89% across related domains. Using a two-step mapping process with the similarity measures *b-substr* and *wikipedia* (*2nd mapping*) further increases the results across related domains by more than 13% without having an influence on the results in a single domain. Finally, if related terms are additionally learned (*2nd mapping+learning*), the average F-measure for single domains slightly decreases by 2.5%, however the average F-measure across related domains dramatically increases by 60%.

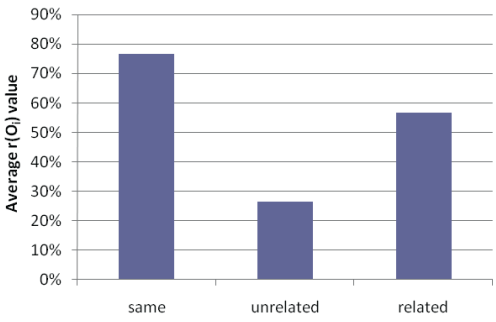


Fig. 8: Average $r(O_i)$ for mapping a context object to a UI from the same, from an unrelated or from a related domain.

C. Distinguishing context objects

Beside determining a good mapping for relevant context objects, it is also important that the relevance values $r(O_i)$ for irrelevant context objects are rather low, so that a context-aware UI is able to distinguish relevant from irrelevant context objects. This means that in the ideal case the mapping process does not assign any context element of an irrelevant context object to a UI element. To evaluate this behavior for our proposed mapping process, we compare the average $r(O_i)$ value for a context object and a UI from the *same* domain, from *related* domains and from *unrelated* domains (i.e. *hotels&identity* etc.). As Figure 8 shows, the average $r(O_i)$ value for the *same* domain (76%) is three times as high as for *unrelated* domains (27%). The average value for *unrelated* domains is comparably high as there is quite a few information in the two domains that requires similar information, e.g. a city name needs to be entered for stating the desired location of a hotel and the address information. This accounts for about 10-20% of the erroneous assignments. The results for the *related* domain (57%) is in between the two extremes. This shows that it is unlikely that a context-aware UI using the proposed mapping process will suggest an irrelevant context object to the user.

VII. Conclusion And Discussion

In this paper, we presented a novel approach for applying similarity measures in context-aware UIs. We showed that the quality of mapping context elements to UI elements can be dramatically increased by applying a second mapping step. Due to practical reasons it can however be of advantage to use only string-based measures as they require far less resources than semantic measures: Semantic similarity measures require an additional knowledge base which requires about 20MB up to several GB of disk space (Wordnet: 23MB, Wiktionary: 42MB, Wikipedia: 6.6GB) and are much slower than string-based measures. Thus, when applying the presented process for example on mobile devices, it is of advantage to rely only string-based measures or to use semantic measures only for a single mapping step.

In any case, learning related terms from observing the user's interactions further improves the quality of the mapping. Further, we demonstrated that taking into account as much information as possible about a UI element or a context element clearly outperforms standard approaches that only rely on the element's label. Finally, we showed that the proposed mapping process is able to distinguish well between context objects that are relevant for a given UI and those that are irrelevant. Thus, the presented mapping process is very well suited for application in a context-aware UI.

ACKNOWLEDGMENTS

We would like to thank SAP Research Darmstadt for supporting our research in the AUGUR project and Torsten Zesch from the Ubiquitous Knowledge Processing Group in Darmstadt for his advice regarding similarity measures.

REFERENCES

[1] E. Rukzio, C. Noda, A. De Luca, J. Hamard, and F. Coskun, "Automatic form filling on mobile devices," *Pervasive Mob. Comput.*, vol. 4, no. 2, pp. 161–181, 2008.

[2] M. Hartmann, T. Zesch, M. Mühlhäuser, and I. Gurevych, "Using similarity measures for context-aware user interfaces," in *Proceedings of ICSC*. IEEE, 2008.

[3] Y. Kalfoglou and M. Schorlemmer, "Ontology mapping: The state of the art," in *Semantic Interoperability and Integration*, ser. Dagstuhl Seminar Proceedings, no. 04391, 2005.

[4] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *VLDB Journal: Very Large Data Bases*, vol. 10, no. 4, pp. 334–350, 2001. [Online]. Available: `http://citeseer.ist.psu.edu/rahm01survey.html`

[5] W. Wu, C. Yu, A. Doan, and W. Meng, "An interactive clustering-based approach to integrating source query interfaces on the deep web," in *Proceedings of SIGMOD*, 2004, pp. 95–106.

[6] J. Stylos, B. A. Myers, and A. Faulring, "Citrine: providing intelligent copy-and-paste," in *Proceedings of UIST*, 2004, pp. 185–188.

[7] T. Chusho, K. Fujiwara, and K. Minamitani, "Automatic filling in a form by an agent for web applications," *Proceedings of 9th Asia-Pacific Software Engineering Conference*, 2002.

[8] C. Fellbaum, *WordNet An Electronic Lexical Database*, C. Fellbaum, Ed. Cambridge, MA: MIT Press, 1998.

[9] M. A. Jaro, "Probabilistic linkage of large public health data file," *Statistics in Medicine*, vol. 14, pp. 491–498, 1995.

[10] W. E. Winkler and Y. Thibaudeau, "An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census," Washington, D.C., Statistical Research Report Series RR91/09, 1991.

[11] A. E. Monge and C. P. Elkan, "The field matching problem: Algorithms and applications." in *Proceedings of KDD*, 1996.

[12] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records," in *Proceedings of KDD*, 2003.

[13] Y. Qiu and H. Frei, "Concept Based Query Expansion," in *Proceedings of the ACM International Conference on Research and Development in Information Retrieval*, 1993.

[14] T. Zesch, C. Müller, and I. Gurevych, "Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary," in *Proceedings of LREC*, 2008.

[15] H. Schmid, "Probabilistic Part-of-Speech Tagging Using Decision Trees," in *International Conference on New Methods in Language Processing*, 1995.

[16] "The UIUC web integration repository," Computer Science Department, University of Illinois at Urbana-Champaign. `http://metaquerier.cs.uiuc.edu/repository`, 2003.