Ontology-based Situation Recognition for Context-Aware Systems[*]

Judie Attard, Simon Scerri, Ismael Rivera, Siegfried Handschuh
Digital Enterprise Research Institute
National University of Ireland
Galway, Ireland
name.surname@deri.org

ABSTRACT

Today's personal devices provide a stream of information which, if processed adequately, can provide a better insight into their owner's current activities, environment, location, etc. In treating these devices as part of a personal sensor network, we exploit raw and interpreted context information in order to enable the automatic recognition of personal recurring situations. An ontology-based graph matching technique continuously compares a person's 'live context', with all previously-stored situations, both of which are represented as an instance of the DCON Context Ontology. Whereas each situation corresponds to an adaptive DCON instance, initially marked by a person and gradually characterised over time, the live context representation is constantly updated with mashed-up context information streaming in from various personal sensors. In this paper we present the matching technique employed to enable automatic situation recognition, and an experiment to evaluate its performance based on real users and their perceived context data.

Categories and Subject Descriptors

The Web of Data [**Vocabularies, taxonomies and schemas for the web of data, Location-based services and mobile semantic applications, Linked sensor data and machine-to-machine communication**]

General Terms

Ontologies, Context Matching, Context Awareness

1. INTRODUCTION

Last-generation personal devices generate a sufficiently large and comprehensive amount of context information that, if successfully integrated and interpreted, has the potential of indicating the owner's habitual tasks, activities and actions. Although the term 'context information' is open to many interpretations, we stick to Dey's definition, i.e., any kind of information that can be used to characterise the situation of an entity [4]. In this case, the entity refers to any person using today's technology, devices and online services. More specifically, we differ between two main context-related concepts. A person's *Live Context* refers to their *transient* context, which comprises of information about the documents being accessed, people who are in their vicinity, tasks and projects they are working on, goals they have in mind, environmental data (including weather conditions), and the device(s) they are using [15]. In contrast, a personal *Situation* refers to a *recurring* 'fuzzy' context, which combines physical (i.e., existence of an entity in some particular region of space and time [14]) and virtual (i.e., interpreted existence of an entity through online activities and device and application use) user circumstances that are independent of a particular point in time. Then, the relationship between a situation and a live context is that the former can be 'characterised' through a series of positive and negative examples, each corresponding to live context observations taken at different times. Thus, situations (e.g. "Working at the Galway Office", "Attending Conference", "At the Cinema" and "Business Meeting") refer to multiple live context instances, and can recur multiple times.

The objective of most context-aware systems is to utilise acquired context information in order to provide context-aware support to people [6]. This objective is also shared by the di.me project, which seeks to provide context-aware warnings, suggestions and automation. Other major objectives are to facilitate the management and privacy-aware sharing of information in a person's distributed personal information sphere[1]. The situation recognition component feeds a context-driven rule system [3] that is customisable by the user through an adequate user interface [12]. Depending on the recognised situation, this is able to change device or application settings, e.g., change device mode or online presence message when entering the workplace; change information sharing settings, e.g., only share status messages with close friends while on holidays; and provide suggestions or warnings, e.g., suggest to share an instant photograph with nearby trusted friends, or provide a warning when an undesirable person is nearby.

Although the contributions in this paper are mainly driven by the above project, our situation recognition technique remains independent of it. Its reliance on a machine-interpretable ontology as a standard context representation format, enables the technique to be adopted by any context-aware system. Moreover, generated context data can also be processed by any remote service that is able to query and process RDF data. The Context Ontology (DCON)[2]

[*]This work is supported in part by the European Commission under the Seventh Framework Program FP7/2007-2013 (*digital.me* – ICT-257787) and in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (*Líon-2*).

ISEM '13 September 04 - 06 2013, Graz, Austria
Copyright 2013 ACM 978-1-4503-1972-0/13/09 ...$15.00 http://dx.doi.org/10.1145/2506182.2506197.

[1]Project website: http://dime-project.eu/

[2]http://www.semanticdesktop.org/ontologies/dcon/

is adopted to represent both a person's unique and dynamic live context, as well as their recurring situations. To generate and update a person's live context, we treat each of their devices (including laptops, smartphones and other gadgets) and services (social networks, weather, scheduling and email services) as nodes in a personal sensor network. The storing of a situation is manually initiated by the system's user, whereby the live context instance is persisted and transformed into a situation instance. Gradually, the presented technique will enable the gradual characterisation of situations through more positive and negative examples, requiring minimal supervision by the user. In order to be able to achieve this functionality, we first need a proper mechanism that enables the continuous monitoring of dynamic context information from personal sensors, and its aggregation into higher-level context interpretations. Once this mechanism is provided by a context-aware system, our context matching technique is able to semi-automatically:

1. Identify recurring situations by periodically comparing a 'live context' to previously-identified situations;

2. Suggest matched situations and gradually characterise them based on the user's feedback.

In this paper we describe how we target the above aims to provide an adaptive situation recognition technique. The use of ontologies for context representation, and the ensuing RDF-graph matching technique, are the defining novelties of our approach. 'Raw' and interpreted context information acquired from various types of personal sensors, is merged onto a unified DCON representation that is both portable and machine-processable. This addresses existing limitations of many context-aware systems such as [8], which are neither sufficiently comprehensive to model the wide range of available context sensor data; nor do they allow for its unified representation. Unless scattered context information can be unified and interpreted as a whole, its exploitation to provide context-aware features remains limited. In addition, due to the use of different context models in most context-aware systems, it is not possible to re-use or process the generated rich context information outside its native environment.

In the rest of the paper, we first compare related work in Section 2, before providing the details of our technique in Section 3. In Section 4 we then report and discuss the results of its evaluation. Future work plans are included within the conclusion in Section 5.

2. RELATED WORK

Although there have been many efforts in the area of context-aware systems, we hereby compare our approach to only the ones that are most similar in terms of both final objectives and candidate techniques. SenSay [16] is one example of a context-aware system that attempts to adapt to dynamically-changing environmental and physiological states. The system relies on a sensor network that utilises a person's mobile phone as a primary source of context information, although the use of applications (e.g. calendars, task lists) and body-embedded sensors was also targeted. Inspired by Sensay, Krause et al. [7] implemented a wearable system that can learn context-dependent personal preferences by identifying individual user states and observing user-system interactions during these states. Such systems are comparable to ours in the sense that their main goal is to match a person's current context to a set of predefined states. In comparison, our context-aware system targets more realistic context sensors in order to interpret a person's different 'states'. Our targeted context sources are already available, embedded as physical sensors into personal devices, or available as implemented user activity monitors. Therefore, the retrieval of the required context information is readily available from personal devices and, as explained later, also online sources.

Given that we target a number of existing context sources, each of which generates homogenous data, we looked at context-aware systems that are capable to a given extent of representing it. Moon et al. [8] propose a context-aware system that offers personalised services to the user. Knowledge processing techniques are applied on context information acquired from user ambient sensors, in order to recognise the user's situation and offer relevant services. Although their system utilises an ontology for context knowledge representation purposes, their model is very limited in comparison to the DCON model for various reasons. First, their context model targets a comparatively limited number of context elements (e.g., networks, devices, services, location) than those provided for by DCON. Secondly, DCON supports the representation of both raw context information (e.g. bluetooth network detected), and its interpretation (bluetooth device owner is nearby). Additionally, DCON also organises acquired context information into multiple schema levels (refer to Section 3.1) in order to be able to compare different contexts at different levels.

Once both live context and situations are available as unified DCON instances, i.e., as RDF graphs, we turn our attention to graph-based matching techniques. The latter include both instance/data and schema/structure levels [10]. Given that graph structure in this case has been standardised by DCON, approaches which attempt to discover common substructures between graphs are here irrelevant. However, some commonly-implemented elements of these techniques, e.g. the use of a weighting system to bias the matching process and give prominence to certain important nodes in [5, 1], have been incorporated into ours. Therefore, the success of our context matching technique depends on how able it is to compare two graph instances at the data/content level. SemMF [9] is a semantic matching framework which enables the calculation of semantic similarity between RDF graph representations. It allows for taxonomic and non-taxonomic concept matching techniques to be implemented on particular elements, which can also be assigned varying weights. This made SemMF an instant candidate for our context matching requirements, although a number of limitations were instantly identified.

The most limiting characteristic of SemMF is that it does not differ between node types, and will therefore compare any node in graph *A* against any node in graph *B*. A major adaptation of SemMF was therefore to take advantage of ontology knowledge and only compare items of the same type (e.g. a wifi connection against other wifi connections). Rather than simply identifying common nodes between two context graphs (e.g. a node representing a specific wifi connection), we also want to gauge the extent of their similarity, based on the available metadata (e.g. in one context snapshot, the connection's signal was stronger than in the second snapshot). Similarly, we also want to compare the similarity of non-equivalent nodes of the same type, since a person's situation is not necessarily influenced by specific items, but by the presence of any such items in general (e.g. the presence of *any* person in the vicinity influences the user's situation). Therefore, in order to achieve a more representative weighted average of nodes in any two context graphs, we extended the SemMF metric to perform matching at three schema levels. In fact, our metric does not only consider available nodes, but also their category and more importantly their attributes. Here, we perform constraint-based matching on any two nodes of the same type (e.g., type person), to determine their similarity. Constraint-based matching is driven by the ranges specified for each DCON context attribute (e.g., wifi signal can be between 0 and 5, inclusive). Another extension to SemMF's matching metric involves the use of different weighting systems for each of the three schema levels being matched.

3. SITUATION RECOGNITION

In this section we describe the techniques underlying our situation recognition component, starting with an introduction of DCON as a context representation schema, a description of the employed context information mining techniques, the comparison of context graphs, and the semi-automatic situation adjustment system. Before going into the details, we first explain our general approach towards handling and interpreting context information. Specifically, we differ between three types of context information:

a) 'raw' context information that is directly retrieved from sensors (e.g. specific location, time, running applications, speed and movement, environmental factors);

b) 'interpreted' context information derived from raw context (e.g. detection of nearby people based on device proximity and network connections, place types such as 'food establishment' or 'workplace', time of the day/year such as 'morning' or 'evening', activities such as 'sleeping' or 'working');

c) 'situations', recurring combinations of raw and interpreted context patterns that mean something to an individual (e.g. "working@office", "Attending Conference", "Travelling to the UK").

Raw context is retrieved from a personal sensor network composed of various devices and services. In di.me, the covered devices include personal computers, laptops, smartphones and other gadgets (e.g. Fitbit[3]). Personal services include mainstream social networks (such as Facebook, Twitter, LinkedIn and Google++), iCal-compliant calendars and weather services. Raw context perceived through this sensor network includes active network connections (e.g. cellular, wired and wireless networks), physical user environment (e.g. noise and brightness levels, accelerometer data, current and forecast weather), device activity (e.g. key stroke, cursor, running applications, open files), commitments (current and upcoming calendar events), locations and nearby people (social network check-ins and person tags), online presence (from IM applications).

Additional user activity context can be interpreted from the retrieved sensor data. In di.me, examples of interpreted context include person(s) proximity detection via network connections, place types based on the user's own tags and publicly available maps (e.g. 'home', 'workplace', 'restaurant', 'sports centre', 'airport'). More sophisticated techniques are used to infer user activities (e.g. working, eating, travelling) from either the discovered place type (e.g. restaurant: 'eating'), information extraction techniques to analyse social network status messages (e.g. "On my way to the Airport": 'travelling'), and known associations between certain applications/files being used or edited on either of the devices (e.g. application Eclipse in foreground: 'working'). In addition, discrete data retrieved from sensors (e.g. 23°C, 15:35pm, 54dB) can also be transformed into predefined fuzzy classifications (e.g. temperature: 'warm', time period: 'late afternoon', noise level: 'quiet').

Although both a Live Context and a Situation are composed of a combination of raw and interpreted context elements, the former consists of a single and continuously updated context snapshot, each of which is bound to a specific point in time. In contrast, a situation can refer to multiple live context snapshots, each occurring at different times. Therefore, a situation can be expressed in terms of different combinations of raw and interpreted context elements, organised within one or more live context snapshots.

[3]www.fitbit.com/

3.1 Context Representation

The DCON Context Ontology enables a comprehensive representation of an entity's context, as acquired from device and user activity sensors [11]. DCON context instances are each stored as an RDF named graph [2]. DCON separates between three different layers of context abstractions, based on a model provided by Schwarz [15]. As shown in Fig. 1 (grey box 'A') information corresponding to a *Context* is categorised under different kinds of *Aspects*, each of which consists of various *Elements*, which in turn posses various context *Attributes*. In addition, since we retrieve context information from multiple devices simultaneously, this might result in multiple *Observations* for the same context element (e.g., a wifi is detected by both a person's laptop and their smartphone). Thus, if $A = \{a_1, a_2.., a_m\}$ is the set of all possible DCON aspects, $E = \{e_1, e_2.., e_p\}$ is the set of all possible DCON elements, $T = \{t_1, t_2.., t_q\}$ is the set of all possible DCON attributes; where $m, p, q \in \mathbb{N}$, then:

DEFINITION 1.

$$c \subseteq A,$$
$$a \subset E,$$
$$e \subset T;$$

such that: a context $c = \{a_1, a_2, .., a_r\}$, for each aspect $a \in c$, $a = \{e_1, e_2, .., e_u\}$, and for each element $e \in a$, $e = \{t_1, t_2, .., t_v\}$; where $r, u, v \in \mathbb{N}$ and $r \leq m$, $u < p$ and $v < q$. In addition, each element e can have one or more observations o such that: $e = \{o_1, o_2, .., o_k\}$, $k \in \mathbb{N}$, $k \geq 1$, such that, for each observation $o \in e$:

DEFINITION 2.

$$o = \mathcal{P}(t) \setminus \{\emptyset\};$$

i.e., the context attributes will not be attached to the element. Rather, the element's observation(s) will each carry a non-empty subset of the corresponding element attributes.

Aspects (grey box 'B'), which in the latest DCON ontology number 7 (i.e., in the definitions above, m = 7) refer to the user's: schedule (groups known events and tasks as its elements), peers (groups nearby contacts and groups), environment (various sensor data, weather conditions), attention (user device activities), spatial/temporal data (location, time period), connectivity (devices, networks) and states (physical activities, availability). Each of the embedded elements, which in the current DCON number 20 (p = 20), can be further described using domain ontologies, such as the Presence Ontology (DPO)[4] and the Information Element Ontology set (NIE)[5], to store e.g., a file's size, a contact's address, a wifi's mac address, etc. However, context-dependent information about these elements is only attached through DCON vocabulary. Therefore, no external ontology vocabulary, beside the element type itself, is taken into consideration by our context matching algorithm. Instead, DCON provides a list of context attributes (numbering 39 in the latest version, i.e., q = 39) that are indicative of an entity's context (e.g. nearby wifi's connection status—*dcon:connected*, the physical distance to nearby locations—*dcon:distance*, etc.). As explained in Definition 2 above, these context attributes will not be attached to the element itself, but to its observations. This is because different devices may experience different context attributes

[4]http://www.semanticdesktop.org/ontologies/dpo/
[5]http://www.semanticdesktop.org/ontologies/nie/

for an element (e.g., a person is nearer to your laptop, than to your phone). Apart from the element-specific context attributes, DCON also provides three generic ones. As shown in Fig. 1-A, *dcon:recordedBy* refers to the device through which an observation has been recorded, whereas *dcon:recordedAt* indicates the observation time. In addition, the *dcon:validity* attribute predefines a span of time during which an observation is assumed to remain relevant (e.g. active applications context changes faster than the outside temperature), and is intended for use by context-aware systems like di.me to indicate the required observation update frequency.

Live Context and Situations are both subclasses of the generic *dcon:Context* class (grey box 'C'). As they consequently have the same structure, this simplifies our context matching task. However, a *dcon:Situation* represents a recurrent context abstraction that can be characterised by one or more past *dcon:LiveContext* snapshots, each of which can be either a negative (*dcon:negativeInstance*) or positive (*dcon:positiveInstance*) example of the *dcon:Situation*. In this way, situations are gradually 'trained' in order to achieve a better representation. Positive and negative examples can be persisted as instances of *duho:ContextLog*[6].

Apart from the specific types of context aspects, elements and their attributes, DCON also provides generic vocabulary that can be used to optimise the context matching task (top-left corner of Fig. 1). The *dcon:weight* property can be used to bias certain aspects and/or elements in relation to a situation (e.g. the Spatial-Temporal aspect as a whole is more indicative of a situation, or the presence of a specific person reduces the likelihood of a situation being active). In addition, DCON also provides two special markers that define which elements are necessary for recognising a particular context/situation (*dcon:isRequired*), and the inverse, i.e. its presence would right away eliminate a particular situation from being matched (*dcon:isExcluder*). These properties have the effect of pre-filtering the candidate situations that are to be matched.

```
@prefix dcon: <http://www.semanticdesktop.org/ontologies
     /2011/10/05/dcon#> .
#Named graph containing situation "Working@Office"
<urn:juan:graph:situation14> {

#Relevant Weighted Aspects for this Situation
juan:Situation14Peers a dcon:Peers;
    dcon:weight "0.6";
    dcon:nearbyPeer juan:Person9,
    #Additional people detected in situation...

juan:Situation14Connection a dcon:Connectivity;
    dcon:weight "0.9";
    dcon:connection juan:Connection2.

juan:Situation14SpaTem a dcon:SpaTem;
    dcon:weight "1";
    dcon:currentPlace juan:Location14.
#Other Aspects & Descriptions...

#Weighted Context Elements and their Attributes
juan:Location14 a dcon:Element;
    dcon:weight "1";
    dcon:isRequired "true";
    dcon:hasObservation juan:Observation45.

juan:Observation45 a dcon:Observation;
    dcon:latitude "53.289997";
    dcon:longitude "-9.074094";
    dcon:recordedBy juan:Device1.

juan:Connection2 a dcon:Element;
    dcon:weight "0.7";
    dcon:hasObservation juan:Observation56,
                        juan:Observation57.

juan:Observation56 a dcon:Observation;
    dcon:recordedBy  juan:Device1;
    dcon:connected "true";
    dcon:networkSpeed "11.16";
    dcon:signal "5".

juan:Observation57 a dcon:Observation;
    dcon:recordedBy  juan:Device2;
    dcon:connected "false";
    dcon:networkSpeed "5.8";
    dcon:signal "2".
#Other Elements & Descriptions... }

#Named graph containing metadata about the situation
     graph above
<urn:juan:graph:situation14/metadata> {

#Situation graph descriptions
<urn:juan:graph:situation14> a dcon:Situation;
    nao:lastModified "2012-05-05T00:00:00Z";
    nao:prefLabel "Working@Office";
    dcon:positiveInstance <urn:juan:graph:contextlog43>,
                          <urn:juan:graph:contextlog98>,
                          <urn:juan:graph:contextlog14>;
    dcon:negativeInstance <urn:juan:graph:contextlog24>.}
```

Listing 1: A *dcon:Situation* instance

An example of a DCON Situation representation is shown in the first (urn:juan:graph:situation14) of the two named graphs in Listing 1. Descriptive metadata for the situation (including the user-defined label), as well as references to positive and negative live context examples, are stored in the second graph (urn:juan:graph:situation14/metadata). The situation graph itself only contains the set elements that characterise it, derived from both positive and negative examples, and grouped by aspect. Depending on their type, the element descriptions entail a number of context attributes. Both aspects and elements are assigned weights that bias the situation towards specific context items, e.g. 'Location14' is marked as being necessary for this situation to be even considered as a matching candidate. The observations enable the distinction between the provenance of the attribute data, e.g. two Observations of 'Connection2'; 'Observation56' and 'Observation57' are recorded by 'Device1' and 'Device2' respectively.

3.2 Identifying Recurring Situations

In order to enable the automatic recognition of recurring situations, we require a mechanism for matching the continuously updated Live Context against any number of stored situations. The job of the context matching process is to compare the live context graph contents to all stored situations in order to find the highest degree of similarity between the contained elements[7]. In order to compute the degree of similarity between two context graphs, we adapt and extend the SemMF matching framework to be able to take as input two DCON instances and compare their contents at the three schema levels. We adopt a top-down approach, whereby we start by comparing each available context aspect (e.g. Connectivity) in the life context graph, against the same aspect in the candidate situation. Then, for each aspect, we obtain all available elements (e.g. detected network connections) from both graphs. A confusion matrix is created to match elements of the same *hasContextElement* sub-property (e.g. *dcon:connection*). As explained in Section 2, although we are able to identify common elements occurring in both

[6]The DUHO User History Ontology enables the logging of past user information, including context: `http://www.semanticdesktop.org/ontologies/duho/`

[7]In di.me we are currently matching Live Contexts at a pre-defined frequency. Future work will include devising a technique which will automatically identify substantial changes in the Live Context and initiate matches with the stored situations.

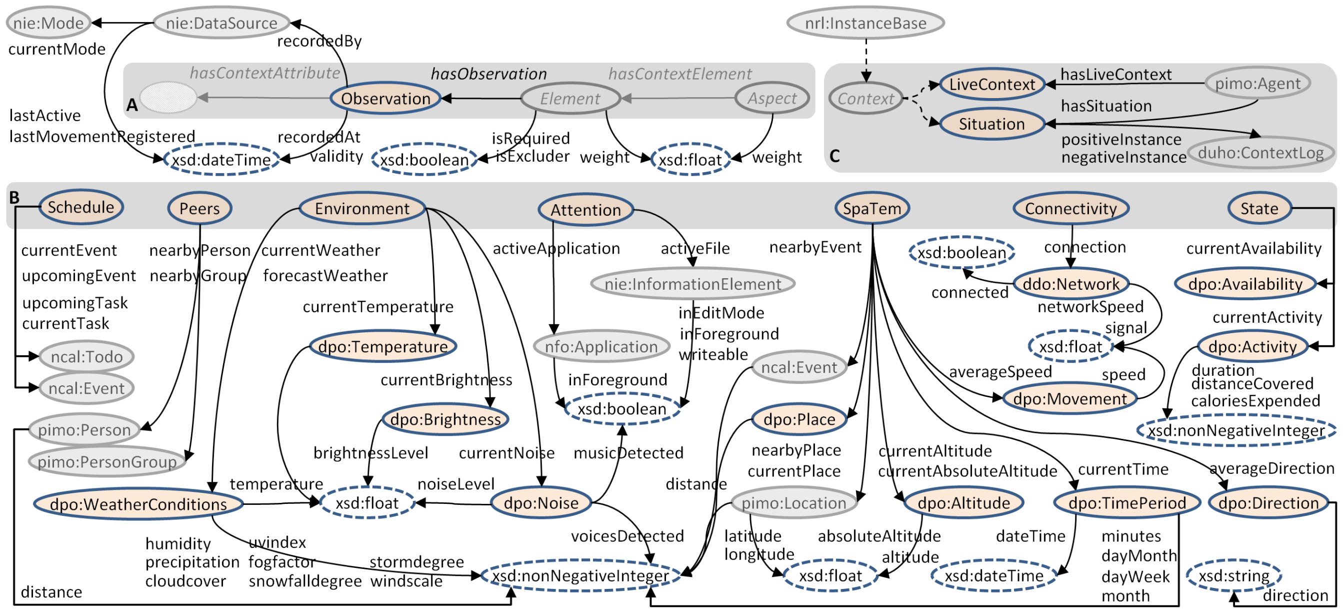
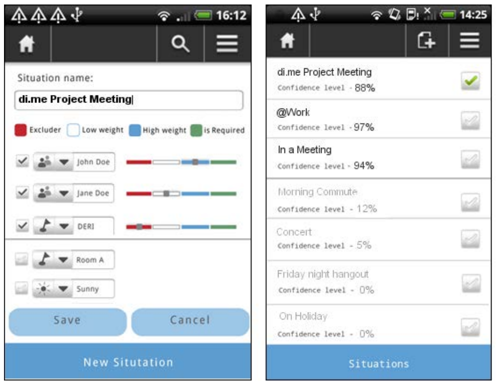


Figure 1: The DCON Context Ontology

graphs (e.g. same wifi network), we also compare non-equivalent elements to gauge their similarity. To this end, we consider the available context attributes for each element, and execute various constraint-based matchers depending on the attribute type and their specified range. Although some of these matchers are trivial (e.g. comparing two *dcon:temperature* values, 20.34 and 18.23, based on the specified datatype and range: $-60 \leq xsd:float \leq 60$), others are more complex (e.g. determining the offset of two locations based on two sets of *dcon:latitude* and *dcon:longitude* values). Through the use of these attribute matchers, we are therefore not only able to detect a similarity between non-equivalent elements (e.g. person A in a graph and person B in a second graph were both detected to be very close to the user, based on the *dcon:distance* attribute), but also to detect dissimilarity between equivalent elements (e.g. a specific wifi network's signal was much stronger in one of the two graphs being matched).

The similarity measure also employs a weighting system which is used to give prominence to context information that is either considered to be more representative of the situation (positive bias), or to decrease the likelihood of that situation having been reactivated (negative bias). Aspects, elements and attributes can all be assigned a numeric weight, although the idea for weight assignment in the di.me system differs for each of the three levels.

The di.me UI, as shown in the prototype in Fig.2 (left), will enable users to (optionally) manually assign weights to detected context elements in order to mark those which they consider to be most representative (high weight) of the situation they are saving. In addition, they can also mark (either at the creation stage or later) those elements which would normally exclude the situation from being a candidate for the automatic situation recognition, as well as those which are required. Further UI details are provided in [13]. Thus Element Weights w_e have a range of: $-1 \leq w_e \leq 1$. Aspect Weights w_a are all equalised at the start of the situation lifecycle, and are only adjusted automatically internally, with a range of $0 \leq w_a \leq 1$. Attributes also have weights, which however are pre-defined by the schema. Attribute weights determine the internal ratio of each attribute for a specific context element, such that some are given more prominence (e.g. a nearby wifi's signal is more indicative of a user's situation than a wifi's connection speed). Thus Attribute Weights w_t have a range of: $0 \leq w_t \leq 1$ such that $\sum_{n=1}^{g}(w_t) = 1$, where g is the number of attributes compared for the element in question, $g \leq v$, and v is the number of possible attributes for an element as defined in Def. 1.


16:12

Situation name:
di.me Project Meeting

🟥 Excluder ☐ Low weight 🟦 High weight 🟩 Is Required

✓ John Doe
✓ Jane Doe
✓ DERI
Room A
Sunny

Save Cancel

New Situtation

14:25

di.me Project Meeting Confidence level - 88%	✓
@Work Confidence level - 97%	
In a Meeting Confidence level - 94%	
Morning Commute Confidence level - 12%	
Concert Confidence level - 5%	
Friday night hangout Confidence level - 0%	
On Holiday Confidence level - 0%	

Situations

Figure 2: Saving Situations and Situation Notification

The context aspect, element and attribute matching process described above, coupled with any corresponding weights, enable us to define a similarity metric that calculates the overall graph similarity between a live context and one or more candidate situations. The similarity score sim_{ls}, calculated for a live context l, against a situation s, is explained below. In this case, to arrive to the definition of sim_{ls} we will start bottom-up from the attribute, observation, element and finally aspect similarity formulas.

Let the similarity between an attribute from l and an attribute from s be sim_t, where the similarity is calculated using the aforementioned attribute matchers. After generating a confusion matrix for comparing observations between two elements of the same *hasContextElement* sub-property, the similarity of one observation each from l and s is calculated as follows:

$$sim_{+o} = 0.5 + (\frac{\sum_{n=1}^{g} (w_t * sim_t)}{g} * 0.5) \tag{1}$$

$$sim_{-o} = 0 + (\frac{\sum_{n=1}^{g} (w_t * sim_t)}{g} * 0.5) \tag{2}$$

where g is the number of attributes compared for the observations in question, $g \leq v$ as defined in Def. 1, w_t is the predefined weight of the attribute, and $n \in \mathbb{N}$. The difference between sim_{+o} and sim_{-o} is that while the former is used when two observations are recorded by the same device, the latter is used in the opposite case. Thus if both observations are recorded by the same device, the similarity starts at 0.5, and is increased depending on the similarity of the attributes. Otherwise, the similarity starts at 0, and is increased up to a maximum of 0.5.

The calculation of the element similarity sim_e is similar to the observation similarity calculation. A confusion matrix is generated to compare elements in l against elements in s, such that elements having the same *hasContextElement* sub-property are compared, for each aspect in turn. For example, for aspect Peers, if l has *dcon:nearbyPerson* Person1 and Person2, while s has *dcon:nearbyPerson* Person2, the latter is compared to both *dcon:nearbyPerson* elements in l. The same applies to nearbyGroup. Thus for aspect Peers, two confusion matrices are generated and we work out the similarity of each pair. At this stage we also differ between equivalent ($+e$) and non-equivalent ($-e$) element pairs. Equivalent pairs indicate that the same element exists in both l and s, following the previous example; *dcon:nearbyPerson* Person2 indicates an equivalent pair. As opposed to equivalent pairs, a non-equivalent pair indicates a distinct pair of elements, following the previous example; the compared pair *dcon:nearbyPerson* Person1 from l and *dcon:nearbyPerson* Person2 from s are non-equivalent. Thus:

$$sim_{+e} = 0.5 + (\frac{\sum_{n=1}^{f} (sim_o)}{f} * 0.5) \tag{3}$$

$$sim_{-e} = 0 + (\frac{\sum_{n=1}^{f} (sim_o)}{f} * 0.5) \tag{4}$$

where f is the number of observations compared for the elements in question, $f \leq k$ as defined in Def. 1, $n \in \mathbb{N}$ and $0.5 \leq sim_{+e} \leq 1$ and $-1 \leq sim_{-e} \leq 0.5$.

Once the similarity of each element in the confusion matrix has been returned, the type similarity sim_{ce} is calculated. This similarity considers all the elements within an aspect which have a common *hasContextElement* sub-property, e.g. *dcon:nearbyPerson* for aspect Peers (refer to Fig. 1).

$$sim_{ce} = \frac{\sum_{n=1}^{x} (w_e * sim_e)}{x} \tag{5}$$

where w_e is the element weight, x is the number of elements compared in an aspect for a single *hasContextElement* sub-property, such that $x \leq u$ as defined in Def. 1 and $n \in \mathbb{N}$. The aspect similarity sim_a is then calculated accordingly:

$$sim_{a} = \frac{\sum_{n=1}^{z} (sim_{ce})}{z} \tag{6}$$

where z is the number of common *hasContextElement* sub-properties of elements in an aspect (e.g. *dcon:nearbyPerson*, *dcon:nearbyGroup* for aspect Peers), within both graphs, $n \in \mathbb{N}$ and $-1 \leq sim_{a} \leq 1$, where a negative aspect signifies that a lot of its matched elements had a negative weight. We are giving all the sub-properties equal weighting, hence the division by z. Once the similarity of each aspect has been worked out, sim_{ls} is computed as follows:

$$sim_{ls} = \frac{\sum_{n=1}^{y} (w_a * sim_a)}{y} \tag{7}$$

where w_a is the aspect weight, $n \in \mathbb{N}$, $-1 \leq sim_{ls} \leq 1$ and y is the number of aspects compared. In order to improve the interpretation of the results, a simple function transforms the results to a value of between 0 and 1.

3.3 Semi-automatic Situation Adjustment

The context matching algorithm is coupled with a feedback loop that enables the situation recognition technique to self-adapt over time, in order to adjust and better characterise represented situations. This functionality requires interaction with the user, as a form of gradual training. Based on the resulting matching scores, the di.me UI will notify the user of possible recurring situations, as shown in the prototype in Fig. 2 (right). The user will also be able to interact with the UI to say whether the suggested situation is really taking place or otherwise. If the user confirms the suggestion, the matched *dcon:LiveContext* instance is taken to be a positive example (*dcon:positiveInstance*) of the situation. If the user rejects the suggestion, the live context is stored as a negative example (*dcon:negativeInstance*). These examples will be used to automatically improve situation representations.

Whenever a new (positive or negative) context example for a situation is logged, the aspect and element weights will automatically and gradually be adjusted to reflect the new example. Context examples improve the situation representation in two ways:

1. by extending the situation with newly observed elements and aspects, having positive/negative initial weights, depending on the nature of the context example (positive/negative);

2. by modifying the weights of existing situation elements that are also observed in the context example, such that they are reduced or increased appropriately given the nature of the example. The current implemented function for the automatic weight reduction/increase is such that it applies a change of an arbitrary value of 0.1[8] with each example, while respecting the bounds specified for w_e in the previous section.

As an example, consider a person being in a social dinner with colleagues. Previously, they have saved a number of situations, including "@BusinessDinner" and "@BusinessLunch". Detecting a number of common elements and their attributes, the context matching component will present both situations to the user, who then confirms that "@BusinessDinner" is representative of their current situation, whereas the other suggestion is rejected. This user feedback will result in the current live context to be stored as a positive example and a negative example, of the confirmed and rejected situations respectively. Both situations are then adjusted so as to better reflect this new knowledge. Here, the system will identify previously unobserved context elements, such as a *dpo:TimePeriod* instance that represents the evening. This element will be introduced under the Spatial-Temporal context aspect (*dcon:currentTime*) for both situations, with initial positive weights for the "@BusinessDinner" situation, and negative weights for "@BusinessLunch". Secondly, the weights of existing items will be adjusted. For example, whereas in previous occurrences of

[8]This value is subject to further experimentation in future work.

the confirmed situation person 'John Doe' was always observed, in the latest positive example the same element is not part of the Peers aspect. From this, one can interpret that this person is not very important for the characterisation of this situation, and therefore its weight is automatically reduced.

4. EVALUATION

In this section we describe and discuss the setup and results of an evaluation of our context matching technique. The evaluation comprises two parts: an experiment to gauge the accuracy of the matching algorithm, and an evaluation of the usefulness of the planned user-suggestion of top-matching situations. The evaluation was particularly challenging due to the incompleteness of the available di.me userware prototype, which is not yet able to provide all the planned UI features. These include the list of suggested situations shown on the right-hand side of Fig. 2, and the feedback loop in Section 3.3. Therefore, to carry out this evaluation, we partially relied on the existing prototype, and partially on an offline execution of the remaining functions.

4.1 Technique Accuracy Evaluation

4.1.1 *Experimental Setup*

For this experiment we required the following setup. Seven persons having intermediate to advanced technical levels (fellow researchers and students) were asked to download the di.me mobile application[9] on their phones and retain it for a total of 2 weeks. During this time, the integrated context extraction service constantly streamed context information to a di.me live context graph. Meanwhile, the evaluators were instructed to identify between 1 and 3 personal situations each. Examples of identified situations, which totaled 14, include 'Working@Office', 'Gym', 'Football', 'Out with friends' and 'Shopping'. During the two weeks, each evaluator was required to mark 4 positive, and at least 4 negative instances of each of these situations in their day-to-day life. In this way, a total of 104 corresponding live contexts were stored. Of these, 56 consisted of positive, and 48 of negative examples of the above situations.

Due to the unavailability of the UI component which executes the feedback loop, in order to characterise the identified situation we proceeded as follows. After the end of the experiment, for each situation we randomly singled out one negative and two positive instances out of the corresponding live contexts stored by the evaluators. The feedback loop algorithm was then executed iteratively for each of these three instances, in order to 'merge' them in a DCON situation. The remaining live context instances for that situation where then each fed to the context matching algorithm, for automatic matching against the generated situation.

4.1.2 *Results and Discussion*

Before the experiment, evaluators 1 to 7 each identified 3, 2, 3, 2, 1, 1 and 2 situations, and a corresponding 19, 16, 23, 16, 8, 8 and 14 live contexts respectively, resulting in 62 distinct live contexts[10]. Following this experiment, we first tried to determine whether our context matching result can deterministically identify whether a situation is recurring, or otherwise. For the purpose, we started on the basis of the score transformation explained at the end of Section 3.2, such that $sim_{ls} > 0.5$ signifies graph similarity, $sim_{ls} < 0.5$ dissimilarity, and $sim_{ls} = 0.5$ being a neutral score. Therefore, our first experiment consisted of comparing the results of manual (as identified by the evaluators) versus automatic situation-live context matches, to an increasing threshold varying from 0.49 to 0.55. The resulting F-measures, weighing recall and precision equally, are shown in Fig.3-1. At a high of only 0.545 (Fig.3-2a), they indicate that our technique is currently not suitable for deterministic situation matching.

[9]This application will be available as a public-beta in June 2013

[10]This data is available online at `http://smile.deri.ie/Context_Matching_Evaluation`

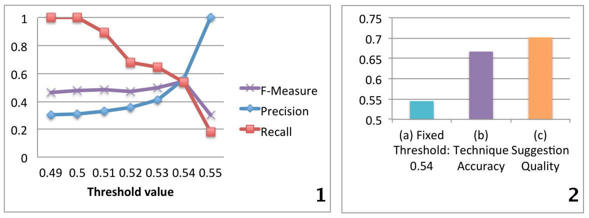


Figure 3: F-Measures obtained when varying threshold (1), and comparing the highest result (2a) to F-measures for Technique Accuracy (2b) and Suggestion Quality (2c)

Next, we wanted to determine whether the context matching technique is able to rank the matching situations correctly, i.e., whether it is useful when the matching results are comparatively interpreted as a set, rather than individually. In this case, rather than relying on a fixed threshold, we identified potential positive matches by calculating the standard deviation for the live context instances of each situation. The results that were above the upper-bounds were then compared to the positive matches identified by the evaluators. An example for evaluator 2 is shown in Fig.4. This evaluator identified 2 situations (sit:2001, sit:2002), the first having 2 corresponding positive (marked 1), 6 negative (marked 0) examples (lc:2001-lc:2008), the second having 2 positive and 5 negative examples. Calculating the standard deviation for sit:2001 ($\sigma = 0.005615283$), the positive matches resulting from the context matcher (shaded light gray) correspond to those identified by the user (true positives). For sit:2002, one of the resulting positive matches (dark gray) did not match the manual identification (false positive). The F-measure for all effected comparisons totaled 0.667 (Fig.3-2b) with 0.882 precision, 0.536 recall. Therefore, this indicates that our context matching algorithm is rather successful for a comparative ranking of context matches.

	sit:2001	
	Auto	User-Marked
lc:2001	0.53277272	1
lc:2002	0.53196013	1
lc:2003	0.518890977	0
lc:2004	0.521435618	0
lc:2005	0.519114196	0
lc:2006	0.526001811	0
lc:2007	0.530018151	0
lc:2008	0.519292772	0

	sit:2002	
	Auto	User-Marked
lc:2009	0.512876153	1
lc:2010	0.507840514	1
lc:2011	0.514256001	0
lc:2012	0.50125432	0
lc:2013	0.498488396	0
lc:2005	0.498342872	0
lc:2008	0.505607605	0

Figure 4: Results of context matching for evaluator 2

4.2 Situation Quality Evaluation

Although the above experiment compares a number of live context instances to each situation individually, in reality the context matcher will compare a single live context (i.e. the current percieved context) to all stored situations. Therefore, the last result does not exactly reflect the expected outcome of the envisaged situation recognition system. For the purpose, we carried out an additional experiment to determine the usefulness of the results, as intended to be shown to the user (Fig. 2). Therefore, in this experiment we perform the reverse experiment: we start with the live contexts that were marked as positive instances of a situation, and compare each of them to all the generated situations collectively, for each user. Since in reality there will always be situations that are ranked on top, the negative live contexts were not considered for this experiment, as we are only interested to see to what extent the suggested situations indeed match a situation when is recurring. The results (Fig.3-2c) show an increased F-measure of 0.702 (precision: 0.69, recall: 0.714), suggesting that the intended application of our context matching algorithm is indeed useful.

5. CONCLUSION

In this paper we presented an ontology-based situation recognition technique that can be employed by any context-aware system that is capable of extracting information corresponding to the concepts supported by the DCON ontology. This ontology is capable of representing context information corresponding to two major concepts: a person's transient Live Context, as continuously updated through their personal sensor network, and recurring personal Situations, which are characterised by one or more previous live context instances. In both cases, context information is stored as RDF Graphs which are matched at multiple levels corresponding to the structure of DCON's context representation, i.e. context aspects, elements and their attributes. Constraint-based matching is performed at the lowest level (attributes), coupled with an ontology-based weighting system. The results are then used to compute similarity scores at the next two levels (elements and aspects), coupled with separate weighting systems which are intended for manual adjustment by human users. Finally, the presented algorithm is then able to take into account the comparisons at these various levels in order to determine the degree of similarity between two context graphs. We have also demonstrated how the context matching technique is extended with an adaptive functionality that sees an improved characterisation of situations over time. The automatic matching technique was evaluated by comparing the results to a gold standard defined and marked by human users. From the various experiments, we conclude that although our technique is not yet suitable to deterministically recognise a recurrent situation, it is sufficiently able to correctly order the highest-matching situations by similarity. This makes it suitable for use in interactive context-aware systems which suggest situation reactivation to their users, like the di.me userware.

Future work will include further experiments on the weighting scheme (including similarity metrics) in order to identify the contribution of context element weights, with the objective of achieving a deterministic threshold for the similarity score. We will also focus our efforts on optimising the weight-adjustment performed by the feedback loop, in order to speed up the situation adjustment. Our target is for the weight adaptation to achieve a logarithmic behaviour, such that the characterisation of situations will initially be more sensitive to the context examples, thus speeding up the adaptation process while ensuring that the adaptation is cautious enough to remain realistic. At the moment, the situation recognition technique relies on the explicit provision of situations in order to perform context matching. We would therefore like to investigate how, and to what degree, this initial stage could also be automated. An exciting future direction is the possibility to apply data mining techniques for the discovery of patterns in the dynamic live context, so as to automatically discover new potential situations and suggest them to the user, thus inching a step forward towards a fully-automised situation recognition system.

6. REFERENCES

[1] D. Baggenstos. *Implementation and evaluation of graph isomorphism algorithms for RDF-Graphs*. 2006.

[2] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th international conference on World Wide Web*, page 613, New York, New York, USA, 2005. ACM Press.

[3] J. Debattista, S. Scerri, I. Rivera, and S. Handschuh. Ontology-based rules for recommender systems. In *Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data*, 2012.

[4] A. K. Dey. Understanding and using context. Technical report, Future Computing Environments Group, Georgia Institute of Technology, Atlanta, GA, USA, 2001.

[5] F. Esposito, D. Malerba, and G. Semeraro. Flexible matching for noisy structural descriptions. In *In proceeding of 12th IJCAI*, pages 658–664, 1991.

[6] J. Hong, E. Suh, and S. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509 – 8522, 2009.

[7] A. Krause, A. Smailagic, and D. P. Siewiorek. Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array. *IEEE Transactions on Mobile Computing*, 5:113–127, 2006.

[8] A. Moon, Y. Park, and S. Kim. Short paper: Situation-awareness model for higher order network knowledge management platform. *Proc. Semantic Sensor Networks*, page 110, 2009.

[9] R. Oldakowski and C. Bizer. SemMF: A Framework for Calculating Semantic Similarity of Objects Represented as RDF Graphs. *Poster at the 4th International Semantic Web Conference (ISWC 2005)*, 2005.

[10] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB JOURNAL*, 2001.

[11] S. Scerri, J. Attard, I. Rivera, M. Valla, and S. Handschuh. Dcon: Interoperable context representation for pervasive environments. In *In Proceedings of the Activity Context Representation Workshop at AAAI 2012*, 2012.

[12] S. Scerri, A. Schuller, I. Rivera, J. Attard, J. Debattista, M. Valla, F. Hermann, and S. Handschuh. Interacting with a context-aware personal information sharing system. In *Proceedings of the 15th International Conference on Human-Computer Interaction (HCI2013) [to appear]*, 2013.

[13] S. Scerri, A. Schuller, I. Rivera, J. Attard, J. Debattista, M. Valla, F. Hermann, and S. Handschuh. Interacting with a context-aware personal information sharing system. In *Proceedings of the 15th International Conference on Human-Computer Interaction (HCI2013) [to appear]*, 2013.

[14] D. W. Schloerb. A quantitative measure of telepresence. *Presence*, 4(1):64–80, 1995.

[15] S. Schwarz. A context model for personal knowledge management applications. In *Modeling and Retrieval of Context, Second International Workshop, MRC 2005, Edinburgh, UK, July 31 - August 1, 2005, Revised Selected Papers*, volume 3946 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2006.

[16] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. L. Wong. Sensay: A context-aware mobile phone. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, ISWC '03, Washington, DC, USA, 2003.