

Architectural Backpropagation Support for Managing Ambiguous Context in Smart Environments

Davy Preuveneers and Yolande Berbers

Department of Computer Science, K.U. Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium,
{davy.preuveneers, yolande.berbers}@cs.kuleuven.be
<http://www.cs.kuleuven.be>

Abstract. The evolution to ubiquitous information and communication networks is evident. Technology is emerging that connects everyday objects and embeds intelligence in our environment. In the Internet of Things, smart objects collect context information from various sources to turn a static environment into a smart and proactive one. Managing the ambiguous nature of context information will be crucial to select relevant information for the tasks at hand. In this paper we present a vector space model that uses context quality parameters to manage context ambiguity and to identify irrelevant context providers. We also discuss backpropagation applied in the network architecture to filter unused context information in the network as close to the source as possible. Experiments show that our contribution not only reduces the amount of useless information a smart object deals with, but also the distribution of unused context information throughout the network architecture.

1 Introduction

The next wave in the era of computing will be outside the realm of the traditional desktop. In the Internet of Things paradigm [1], everything of value will be on the network in one form or another. Everyday objects, such as shoes, cars, coffee cups, refrigerators, bath tubs, will be in the communication range of interconnected devices, and provide relevant context information to create a smart and proactive environment. This context information [2] is considered to be a key ingredient to create a whole range of smart entertainment and business applications that are more supportive to the user.

In a less than ideal world, however, smart objects must be aware of the usability of the imperfect context information that they are provided with [3]. If not, a smart context-aware object may mistakenly assume that the information it retrieves from other smart objects in the network (i.e. the context providers), is accurate and relevant for the purpose of its actions. Therefore, the smart object needs a way to distinguish context providers based on the relevance of the information they provide.

In this paper we first discuss how ambiguous and imperfect context is modeled and verified before usage. This model is based on several information specific context quality parameters that contribute to the diminution of information ambiguity, such as precision and up-to-dateness [4]. Other non-information specific parameters model the reliability of a context provider delivering its information to a particular smart object in the network. Secondly, given these two sets of context quality parameters, we designed an algorithm to identify and ignore invaluable context providers. To further improve the quality of the context information propagated through the network, we developed a backpropagation mechanism to inform the context providers about the usage of their information. This mechanism allows smart objects in the network to selectively redirect information to where it is considered useful. Our technique is compared to context dissemination using network hops as a boundary for propagation. After a brief period of learning, our solution reduces ambiguity by ignoring irrelevant context providers early on, and improves the bandwidth usage ratio of the valuable context information passing through a smart object in the network.

In section 2 we discuss several context quality parameters and show how they are represented in an information model that helps the smart object to decide whether information is relevant for its purposes or not, and to identify (ir)relevant context providers. In section 3 we describe the algorithm to back-propagate the information relevance to the context providers. In section 4 we conduct experiments that illustrate the improvement of the quality of the context information that passes through and is being used by a smart object. Section 5 provides an overview of related work on the distribution of ambiguous context information. We end with conclusions and future work in section 6.

2 Modeling the Ambiguity of Context Information

A smart object may rely on external context providers to take well-informed decisions. Ambiguity arises when multiple parties deliver different information due to the diverse circumstances in which they operate. In this section, we clarify the boundaries that are needed in terms of spatial and temporal relevance, accuracy and precision, and semantic interpretability to understand these situations.

2.1 Disambiguation with Context Quality Parameters

Many aspects of information quality have already been investigated in the last decade [5,6]. In this section, we will only discuss a selection of these information properties that are used as context quality parameters in our model:

1. **Accuracy:** Accuracy refers to the degree of veracity. It describes the closeness of the measured value to the actual true value, or the extent to which the provided information is correct (e.g. $36.8^\circ C$ vs. $37.1^\circ C$, or 99%).
2. **Precision:** This parameter is closely related to the accuracy parameter. It describes how detailed a measurement is stated (e.g. $36.9 \pm 0.1^\circ C$). Measurements that are precise are not necessarily accurate, and vice versa.

```

<ContextItem>
  <ProviderUID>32afc954-a85e-c7b1-f636-123564831237</ProviderUID>
  <Class>http://localhost/context#Temperature</Class>
  <Value type='http://localhost/units#Celsius'>20.6</Value>
  <Precision type='absolute'>0.2</Precision>
  <Accuracy type='relative'>0.95</Accuracy>
  <TimeStamp>2006-09-30 09:37:41+01:00</TimeStamp>
  <Coverage>http://localhost/context#Room</Coverage>
  <Location type='http://localhost/context#Office'>02.45</Location>
</ContextItem>

```

Fig. 1. A *Temperature* information instance of a sensor on the network

3. **Spatial coverage:** This parameter defines the geographic scope of the information. For example, the temperature provided by an outside weather station is unusable for controlling the central heating in a building.
4. **Timeliness:** Timeliness refers to how current the provided information is at the time of delivery. If not sufficiently up-to-date, the temperature information may not be useful either for the task at hand.
5. **Semantic interpretability:** Some context providers may use semantically related terms, e.g. 'Corridor' and 'Hallway'. If these relationships are not understood, any relevant information specified in these terms will be neglected.

The previous parameters help to disambiguate the information itself. The following parameters describe the ability of a context provider to reliably provide useful information to a particular smart object:

1. **Objectivity:** The objectivity property describes the reliability and trustworthiness of the information source. The context provider may have specific intentions to deliver biased information.
2. **Availability:** A smart object is never sure if a context provider will be able to deliver the required information. The context provider may leave the network or its information may be dropped under way.
3. **Completeness:** This parameter reflects the amount of missing or uninterpretable context quality parameters that make the context less reliable without comparison to other available information sources.

These three subjective parameters are not set by the context provider itself, but by the receiving end. They have values between 0 and 1 and are quite likely to be different for each receiving smart object. Fig. 1 illustrates a sample of information delivered by a context provider with a specific *ProviderID*. It provides *Temperature* information using the *Celsius* scale. The *Precision* and *Accuracy* are specified in terms of relative or absolute values. The information is timestamped and the location and coverage are specified as well. Ontologies are used for semantic annotation of the *Class*, *Value*, *Coverage* and *Location* properties. Multiple values can be provided in different scales (e.g. *Celsius* and *Fahrenheit*) or specified semantically (e.g. *Office* and *Room*). For example, the context information in Fig. 1 is not relevant when requesting the *Body* temperature, because an *Office* is not semantically part of a *Body*. These relationships are modeled in an ontology as shown in Fig. 2. Subsumption (*is*-relationship) and semantic distance (*has*-relationship) in the shortest path between two concepts is used

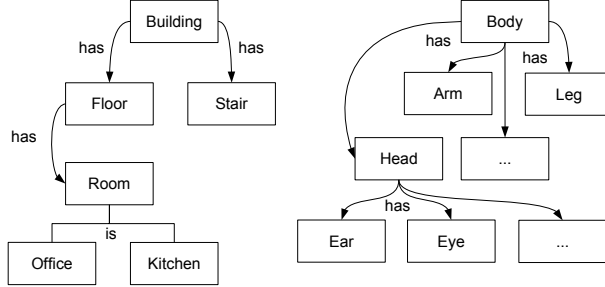


Fig. 2. Ontology specifying semantic containment

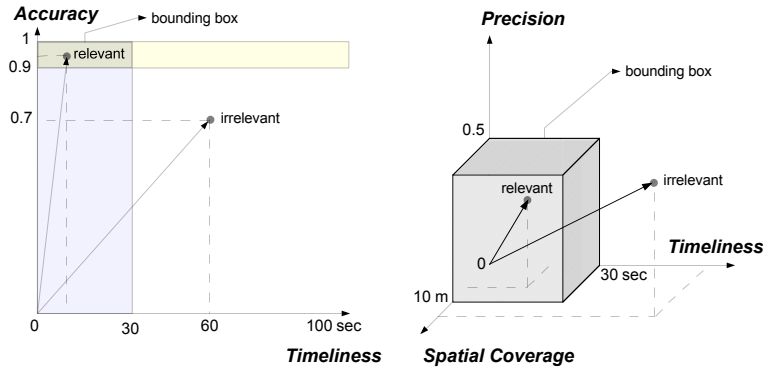


Fig. 3. A relevant and irrelevant context vector in a 2D and 3D vector space model

as a semantic metric for selecting relevant information. For unrelated concepts, the distance is defined infinite (∞). In Fig. 2, the distance between Building and Room is 2. Between Room and Office, the distance is 0 due to the *is*-relationship between the two concepts.

2.2 Modeling Context Quality in a Vector Space Model

In this section, we present a model that represents the context quality parameters in a vector space model to efficiently filter irrelevant context information.

This vector space model allows to explicate the required criteria to which delivered context information should conform. Each dimension of this model represents a context quality parameter, and for each of these dimensions a boundary for acceptable values is defined by the smart object for the task at hand. Together they form a bounding box. Two examples are illustrated in Fig. 3. Numeric parameters are represented as such. Semantic parameters use the semantic metric to represent their value in the model if conversion is not possible. Each context information instance is represented as a vector in the model. In a d -dimensional quality parameter space Q , a context vector \vec{c} is defined as follows:

$$\vec{c} = (c_1, c_2, \dots, c_d) \in Q \quad \text{with} \quad |\vec{c}| = \sqrt{c_1^2 + c_2^2 + \dots + c_d^2}$$

The optimal quality vector \vec{q} represents a context vector that is 100% accurate and precise, up-to-date, and completely relevant for the smart object's purposes. The value of d depends on the number of requirements of the smart object. The relevance $R_{\vec{c}}$ of any context vector \vec{c} can then be measured by computing the weighted distance between the vectors:

$$R_{\vec{c}} = |\vec{q} - \vec{c}| = \sqrt{w_1(q_1 - c_1)^2 + w_2(q_2 - c_2)^2 + \dots + w_d(q_d - c_d)^2}$$

The weights w determine the priority of the quality parameters. The smaller the value of $R_{\vec{c}}$ is, the more relevant the context information is.

2.3 Strategy for Identifying (Ir)relevant Context Providers

A first selection of useful context providers is based on the bounding box described above. The smart object collects incoming information during a certain period of time. All the context vectors are properly aligned (e.g. semantically converted or rescaled to the same type) within the same vector space model.

Given that multiple context providers deliver relevant context information, the actual data should be similar in value as they all comply with a minimum *Accuracy* and *Precision*. Outliers are detected and expunged from the dataset using Grubb's iterative test [7]. For such outliers, including those with undefined quality parameters, the smart object adjusts the subjective context quality parameters (*Objectivity* (O), *Availability* (A) and *Completeness* (C)) for the corresponding context provider. When outliers in the values are detected, the *Objectivity* quality parameter of the context provider P is reduced:

$$O_P \leftarrow O_P \times \text{Requested Relative Accuracy} \quad \text{with} \quad 0 \leq O_P \leq 1$$

If the requested accuracy is low and the context provider claims it complies although its value is an outlier compared to the others, then it is penalized more than if the requested accuracy would have been set higher. If some quality parameters are not provided (e.g. timestamp or location), then its information is more ambiguous and therefore its *Completeness* parameter is reduced. The *Availability* parameter describes how frequent a context provider was able to respond in time. This parameter is crucial when a smart object relies on only 1 provider. Together with the relevance value R_c of the context vector \vec{c} , the three subjective context quality parameters are used to determine an overall score $S_{P, \vec{c}}$ that defines a rank among the context providers:

$$S_{P, \vec{c}} = \frac{R_{\vec{c}}}{O_P \times A_P \times C_P} \geq 0$$

The smart object can then choose to use the value of the context provider with the lowest score $S_{P, \vec{c}}$ or to compute a weighted average value using these scores. The high-level overview of the algorithm is shown below:

Algorithm 1. SortContext(in: info[], out: context[], score[], irrelevant[])

```

1: (BoundingBox box, Vector optimalvector) = ComputeContextQualityBoundaries()
2: for each ContextItem ci in info do
3:   ci.vector.AlignContextVector(box)
4:   if (!box.Contains(ci.vector)) then
5:     ci.Move(info, irrelevant)
6: ContextItem[] outliers = info.ValueOutlierDetection()
7: outliers.UpdateSubjectiveParameters(box)
8: outliers.Move(info, irrelevant)
9: (context, score) = SortRescaledDistance(info, optimalvector)

```

3 Relevance Backpropagation in the Network

Our last contribution in this paper is a backpropagation mechanism to inform context providers in a large-scale and dynamic network architecture about the usage of their information. Proven technologies, such as Bloom filters [8], are used for efficient filtering of the information as close to the source as possible. Intermediate nodes will decide where to forward the information to based on feedback backpropagated by its peers. Whereas the algorithm in a neural network changes the weights of a neuron to converge the error in the output to a local minimum, does our backpropagation algorithm limit the information propagation in the network, possibly impeding the context provider completely from disseminating unused information.

A smart object forwards context information to adjacent smart objects, unless a maximum number of hops is reached. Each forwarding smart object reduces the hop counter, appends its UID to the message delivery chain, and marks the message if the information is relevant for its purposes. Backpropagation to the delivering peer is initiated whenever any of the following conditions arises:

- **Irrelevant context:** The smart object cannot use the context information for its own purposes (see Algorithm 1) and the maximum number of hops is reached or it has no other adjacent peers to forward to.
- **Unused context:** The context information is relevant along the delivery chain, but is not used that frequently. Inform the forwarding peer or the context provider to increase the transmission interval.
- **Duplicate detection:** The smart object has already received the information from another peer through a quicker trajectory. Stop sending the information along this path.

Bloom filters [8] – compact data structures for testing whether an element is part of a set – are used to check if the context information is invaluable for its own purposes and for propagation to adjacent smart objects. Algorithm 2 describes the backpropagation for context information delivery. When a *Duplicate* or *Irrelevant* backpropagation message is received, the smart object adjusts its forwarding filters. For an *Unused* message, the frequency of forwarding is reduced. The receiving peer backpropagates the message further down to the source using the labels in the message delivery chain.

Algorithm 2. BackpropagateRelevance(in: fromPeer, contextMessage)

```

1: (messageRelevant, messageUnused, messageForwarded) = (false, false, false)
2: if (InFilterReceived(contextMessage.ID)) then
3:   BackpropagateMessage(fromPeer, DUPLICATE, contextMessage.ID)
4: else
5:   AddFilterReceived(fromPeer, contextMessage.ID)
6: if (InFilterRelevant(contextMessage)) then
7:   messageRelevant = true
8:   if (InFilterUnused(contextMessage)) then
9:     messageUnused = true
10:    LabelMessage(contextMessage, UNUSED)
11: else
12:   LabelMessage(contextMessage, IRRELEVANT)
13: if (contextMessage.hopsLeft > 0) then
14:   contextMessage.hopsLeft = contextMessage.hopsLeft - 1
15:   for each Peer p in ForwardFilter(adjacentPeers, contextMessage.ID) do
16:     messageForwarded = true
17:     ForwardMessage(p, contextMessage)
18: if (not messageForwarded) then
19:   if (not messageRelevant) then
20:     BackpropagateMessage(fromPeer, IRRELEVANT, contextMessage.ID)
21:   else if (messageUnused) then
22:     BackpropagateMessage(fromPeer, UNUSED, contextMessage.ID)

```

4 Experimental Evaluation

Finding a realistic test scenario of a reasonable size without the detrimental side effects of a real-life network setup is not straightforward, especially for determining the significance of our algorithms on the outcome of the experiments. We therefore chose to simulate and compare the algorithms and mechanisms on an artificial network. The network is generated by means of a weighted location dependent wiring. For k nodes, the first one n_1 is situated in the center of a unit square. The other nodes n_j , $j = 2 \dots k$, are randomly placed in the unit square and connected to at most m existing nodes n_i that each minimize a different function F_m :

$$F_m(n_i, n_j) = H(n_1, n_j) + w_m \cdot D(n_i, n_j)$$

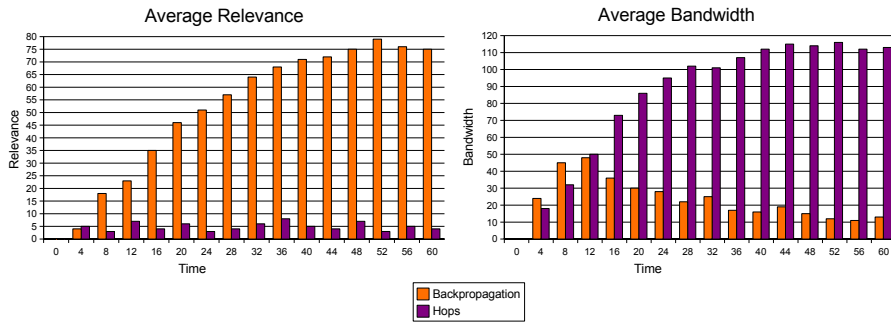
with H = number of hops to node n_1 , D = Euclidean distance, and m different weights w_m = parameter that influences the geographical dependency.

4.1 Experiments

A network is generated with the following parameters: $k = 1000$ nodes, at most $m = 3$ connections with weights $w_1 = 10$, $w_2 = 5$, $w_3 = 1$, and about 100 nodes provide context information of which the characteristics are shown in Table 1. In this smart environment scenario, the *Movement* and *Wireless Body Sensors*, and the *Inhouse Thermometer* make use of the *Clocks* and *Position Beacons*

Table 1. Parametrization of the network architecture

Type	Quantity	Coverage	Update	Hops
Weather Forecast Website	5	1.0	30 min	100
Outside Weather Station	5	0.5	10 min	100
Clock	10	2.0	1 min	50
Movement Sensor	10	0.1	1 min	20
Inhouse Thermometer	10	0.1	5 min	10
Wireless Body Sensor	20	0.05	1 min	5
Position Beacon	40	0.05	2 min	3


Fig. 4. Evolution of context information usage and throughput

to set their own timestamp and coverage. The *Wireless Body Sensor* is used to announce the presence of a particular person. The *Inhouse Thermometers* are used to control the central heating within a specific room related to the person that was detected. The *Clock* is also used to turn on the lights before 8:00 and after 18:00 if movement is detected in the current or nearby room.

The simulation environment assumes that all connections have similar characteristics. In the first experiment, the information is flooded in the network and only limited by the number of hops for forwarding. Filtering and backpropagation as proposed in this paper is applied in a second experiment using the same parameters and network. In Fig. 4 we compare the evolution of the average amount of valuable information for a smart object in the network (with and without backpropagation), and the average network load at a smart object.

4.2 Discussion of the Results

Clearly, the results of this experiment and the graphs in Fig. 4 are only relevant for this particular network parametrization. Nonetheless, for other experiments with different results in terms of absolute values, we could draw similar qualitative conclusions. The left figure shows that the average amount of relevant information (for a peer to which it forwards or for itself) increases up to 75%. This can be improved with larger Bloom filters. They have the characteristic of allowing false positives but no false negatives when testing membership. Bloom

filters cannot handle the deletion of item either when forwarding requirements change. In that case, they are reinitialized which means that less valuable information is again being propagated.

During the initial learning phase, the backpropagation enabled scenario performs slightly worse due to backpropagation messages being sent to the delivering peers, as shown in the right figure. After the learning period, some context providers became inactive as their information was not being used, and others were limited to sending out information into certain directions. The experiment resulted into a significant drop in the overall bandwidth usage in the network.

However, we are aware that the current context propagation is not optimal. In the presence of high speed and low speed connections, it is possible that highly volatile context information is propagated through the slow links, whereas long-lived information is saturating the fast links. As a result, some relevant information might not be considered as our backpropagation mechanism does not take the bandwidth of the network links into account.

The most important advantage is that the smart objects did not need any configuration on where to get their information. They only required connectivity to other smart objects in the network and learned by themselves which context provider was able to deliver relevant information for their purposes.

5 Related Work

Buchholz *et al.* [9] identified parameters that quantify the quality of context and the uncertainty of sensed values. Henricksen *et al.* [10] identified similar imperfection aspects and proposed a graphical model and a software infrastructure for the management and use of imperfect context. In our research, we included extra parameters to explicitly deal with context distribution.

Chalmers *et al.* show in [11] how contextual information can be formulated in the presence of uncertainty using interval arithmetic for uncertain numerical context values. The authors define the *within* and *overlap* relationships to test whether a sensed value range is within a test range and to what degree two values overlap. Our work reuses the same idea in multiple dimensions, but their tree based approach differs from our semantic metric for abstract values.

Dey *et al.* [3] suggest to leverage off techniques such as Bayesian networks and neural networks, as these approaches cannot remove all the ambiguity of sensed information. The authors propose to involve mobile users in aware-environments for the refinement of imperfect context and for reducing ambiguity in context information through a process called mediation to deal with context conflicts. We envision that for large-scale networks this approach will be impractical.

6 Conclusions

In this paper, we have discussed several context quality parameters that affect the ambiguity and uncertainty of context information. We have shown how they can be represented in a multi-dimensional vector space model to allow a smart

object to quickly decide whether any delivered context information is relevant for its purposes or not. We also discussed backpropagation support to provide feedback about the usefulness of the information for any of the smart objects in the network to the respective context providers. We have conducted experiments that illustrate that the amount of information that is propagated through the network but not needed by any of the smart objects on its path is significantly lower than for hop limited context propagation.

Part of our future work, is to take the timeliness of the context information into account to improve the usage of the network capacity. The goal is to achieve just in time context information delivery to the requesting smart objects. Further research will investigate how the bounding box model with its sharp boundaries can be improved by using a normal distribution of the requested context quality parameters, and how the efficiency of the relevance testing is affected.

References

1. International Telecommunication Union: ITU Internet Reports 2005: The Internet of Things 2005, 7th edition (2005)
2. Dey, A.K.: Understanding and Using Context. *Personal Ubiquitous Comput.* 5, 4–7 (2001)
3. Dey, A., Mankoff, J., Abowd, G., Carter, S.: Distributed mediation of ambiguous context in aware environments. In: *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pp. 121–130. ACM Press, New York (2002)
4. Preuveneers, D., Berbers, Y.: Quality Extensions and Uncertainty Handling for Context Ontologies. In: Shvaiko, P., Euzenat, J., Léger, A., McGuinness, D.L., Wache, H., eds.: *Proceedings of Context and Ontologies: Theory Practice and Applications (C&O, Riva del Garda, Italy (2006) pp. 62–64 (2006)*
5. Wand, Y., Wang, R.Y.: Anchoring data quality dimensions in ontological foundations. *Commun. ACM* 39, 86–95 (1996)
6. Lee, Y.W., Strong, D.M., Kahn, B.K., Wang, R.Y.: AIMQ: a methodology for information quality assessment. *Inf. Manage.* 40, 133–146 (2002)
7. Hodge, V., Austin, J.: A Survey of Outlier Detection Methodologies. *Artif. Intell. Rev.* 22, 85–126 (2004)
8. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 422–426 (1970)
9. Buchholz, T., Kupper, A., Schiffers, M.: Quality of Context: What it is and why we need it. In: *Proceedings of the 10th Workshop of the OpenView University Association: OVUA'03, Geneva, Switzerland (2003)*
10. Henricksen, K., Indulska, J.: Modelling and Using Imperfect Context Information. In: *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, Washington, DC, USA, p. 33. IEEE Computer Society, Los Alamitos (2004)*
11. Chalmers, D., Dulay, N., Sloman, M.: Towards Reasoning About Context in the Presence of Uncertainty. In: Davies, N., Mynatt, E.D., Siio, I. (eds.) *UbiComp 2004. LNCS, vol. 3205, Springer, Heidelberg (2004)*