Context-adaptation based on Ontologies and Spreading Activation

Tim Hussein, Daniel Westheide, Jürgen Ziegler
University of Duisburg-Essen
Lotharstr. 65, 47057 Duisburg, Germany
{hussein,westheide,ziegler}@interactivesystems.info

Abstract

Ontologies and spreading activation are known terms within the scope of information retrieval. In this paper we introduce SPREADR, an integrated adaptation mechanism for web applications that uses ontologies for representing the application domain as well as context information like location, user history and local time. Those context factors can be modeled in an ontology and be linked to certain domain nodes. In each session a *Spreading Activation Network* is build based on those ontologies and recognized context factors or user actions can trigger an activation flow through this network. A node's resulting activation value then represents its importance according to the current circumstances. While identically in structure, the Spreading Activation Networks are personalized by automatically modifying link weights and activation levels of nodes. As a result the system learns about the user preferences and can adjust its adaptation mechanism for future runs through implicit feedback.

1 Introduction

The increasing amount of information presented in current web based applications often makes them difficult to use. Finding the appropriate content within the flood of data can be challenging and may cause a user to reject a web application. Various approaches have been proposed to overcome these problems by adaptation, each with its particular advantages and drawbacks. Roughly, two approaches can be distinguished: Context-aware and self-adaptive systems. Context-aware systems adapt the content according to the current circumstances, but usually have no learning mechanism. Ideally a system should continuously observe the user's behavior in order to learn from his decisions and thereby improve future adaptations. This is what self-adaptation is about.

Like [Dey *et al.*, 2004] we propose an integrated view of context and domain information to contextually offer the appropriate content. For this purpose we make use of two concepts originally known within the scope of artificial intelligence and information retrieval: Ontologies and spreading activation.

In Section 2 we present existing approaches that cover context engineering and adaptation techniques. Subsequently, we illustrate our context engineering approach in Section 3 and show a method to integrate contextual information into an existing domain ontology. We propose a method for creating an integrated ontological model for each user representing the current usage context. A modified spreading activation algorithm is then used to adjust those networks according to the user interaction and to refine his profile over time. Simultaneously, self-adaptation is performed by adjusting the adaptation mechanism itself due to implicit relevance feedback. This technique is explained in Section 4, followed by an outline of the system architecture in Section 5 and a description of a prototypical implementation and its evaluation in Section 6. We conclude the article in Section 7, summarizing our conclusions and pointing out future directions for research.

2 Related work

Various authors such as [Middleton *et al.*, 2004] proposed that adaptation by continuous observation is desirable in order to learn from the user's behaviour and the circumstances under which this behaviour occurs. While user modeling and recommendation techniques have been the focus of research for a long time, there have been few attempts that take contextual information into account for purposes of personalization ([Herlocker and Konstan, 2001], [Adomavicius and Tuzhilin, 2001] and [Kovacs and Ueno, 2006]). Current systems mostly concentrate on the user's transaction history, which is of course an important factor for adaptation. Those systems are usually called *user adaptive*. Within applications that are always used in the same context, this is not a problem at all, but with increasing complexity of web applications context becomes a substantial factor in terms of usability.

It can be assumed that there is always a contextual background for the user's information and service needs. Most of the time the circumstances have a crucial impact on our decisions as we always act in different roles. Winter coats might be interesting in November, but not in July. One would only like to know about the cafeteria menu of the day on working days. So a system should automatically adapt itself to the particular context to present the user "the right thing at the right time in the right way" [Kappel *et al.*, 2003]. Those systems are not *user adaptive* but *context adaptive*. Traditional context-aware scenarios focus on single context factors like the user's current location [Chen and Kotz, 2000], for instance for presenting tourist information [Cheverst *et al.*, 2000]. [Henricksen and Indulska, 2005] calls the field of context-aware computing "immature". Though this is a hard judgement we agree that context-adaptivity still has a long way to go. Especially there is a lack of approaches that take the user history into account as well as multidimensional context information.

Recent activities include the *a CAPella* system [Dey *et* *al.*, 2004] which can be "trained" by the user to automatically recognize certain events depending on the current context via multi-modal sensing: Information obtained from a microphone, a camera, RFID antennas and other devices is being used and interpreted. As a result, *a CAPella* for instance recognizes the start of a meeting and automatically presents certain documents that have been used in a similar context in the past. This interesting and promising approach is a good example for sensing and unifying context information. However it strongly focusses on real-world interaction and needs certain external equipment for context sensing, which makes it not directly applicable for the purpose of web engineering.

Collaborative Filtering techniques – well known from the field of recommender systems – have some qualities that make them a good choice for filtering semantically untagged items. This approach is very popular, because it leads to notable results especially in recommender systems and frees the web engineer from creating and maintaining complex content tagging. Yet pure collaborative filtering has some crucial disadvantages, too, like the *ramp-up-problem* often referred to as explained in [Burke, 2002]. Combining collaborative filtering techniques with content based mechanisms has been shown to be a feasible solution to the ramp-up-problem. In [Claypool *et al.*, 1999] and [Melville *et al.*, 2002] different approaches are presented. However, these solutions are rather user centered and do not adequately take the particular context into account. Nonetheless, collaborative filtering algorithms have been proven to be a simple but effective instrument that can be integrated into more sophisticated systems. We think that there is a strong need for an integrated strategy that considers as much context information as possible: The current usage context in terms of situation-awareness as well as the past interactions including the contexts for the time being.

Ontologies have been proven to be a good choice for knowledge representation. Having its roots in philosophy, ontologies have become popular for computer science since the 1990s [Neches *et al.*, 1991]. Ontologies can be used to represent manifold information in a human-understandable and machine-readable format consisting of entities, attributes, relationships and axioms [Guarino and Giaretta, 1995]. Examples for using ontologies in recommender systems can be found in [Middleton *et al.*, 2004], where *Quickstep* and *Foxtrot* are illustrated – two systems that make use of ontologies to recommend scientific research papers a particular user might be interested in. We think that in a truly adaptive system the adaptation techniques have to become part of an optimization process itself through learning mechanisms. So the challenge is to close the gap between user adaptive and context adaptive systems [Oppermann, 2005] and to provide in this sense a holistic system with appropriate learning mechanisms. In this paper we want to introduce SPREADR (Spreading Activation Driven Reasoning) - a model-driven, user- and context-adaptive solution for this problem.

3 Context engineering

Developing a web application with SPREADR implies the creation of several models. A typical scenario is presented in Section 6. The models that can be defined in SPREADR for instance include a domain and a context model. Each of these models plays a distinct role in building the content presented to the user and is an ontology represented in the Web Ontology Language (OWL) format.

Within an ontology, nodes are semantically related to each other. As an extension of this basic mechanism, we assign individual weights to the relations. The resulting networks are identical in structure for each user, but the weights of the links are individualized. Nodes in the ontology (concepts and instances) are treated in a similar way by assigning activation values to them. A fact in reality is valid for everyone, but it is more or less relevant for a certain user. If one considers an item to be important it receives a higher activation value. Thereby, we can create individual weighted networks for each user to represent his personal interests. Information regarding his current location, device, local time etc. can be handled just as well: By raising the activation of the corresponding nodes we can represent an individual usage context and by individualizing the relation weights certain factors are more or less tied to a certain concept for each user.

The domain model
Generally speaking, the purpose of the domain model is to represent knowledge relevant to the respective application as well as pieces of content or references to them, depending on the type of content that is represented [Kaltz, 2006]. To represent relevant domain knowledge, the domain model contains classes and semantic relations between them. Instances of those classes are likewise part of the domain model, linked to each other by appropriate relations. A domain model should be created by domain experts without considering the context-adaptations that are supposed to take place [Kaltz, 2006].

The context model
When building a model-driven context-aware web application, it is not only necessary to model the domain, but also the context. Here, context is "any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves" [Dey, 2001].

Thus, the quality of a context-aware web application depends on the relevance of the modeled context. In order to reduce complexity, we distinguish between five categories of context, as proposed by [Ziegler *et al.*, 2005]:

- *User and role:* individual users or groups of users that are defined according to their different roles.
- *Task:* task-oriented context, e.g. work assignments or a user's personal goals.
- *Location:* the user's physical or virtual location (e.g. internet vs. local area network).
- *Time:* e.g. the season, the weekday or the time of day.
- *Device:* The user's device, e.g a PDA, a mobile phone or a personal computer.

Each of the these context categories is modeled in an ontology containing all context factors the system is supposed to be aware of. At runtime, contextual information is recognized (Section 5) and the respective context factors are activated.

However, for the context to have an effect on the content selection, it is also necessary to model appropriate *context relations*. A context relation defines a link between a context factor and an item from the domain model as well as a relevance weight. This relevance weight is used to determine the general importance of specific domain items in a given context.

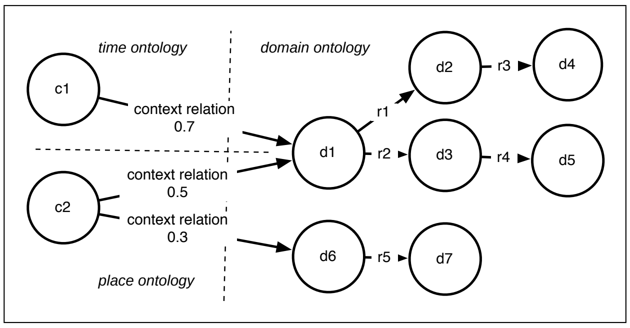


Figure 1: Connecting context factors to the domain model

Usually, when modeling a context relation, it is the context engineer's task to define a relevance weight that reflects the importance of the relation as seen by the majority of users. Therefore, profound knowledge about the target group of the web application is required so as to satisfy the needs of most of the users. Especially when building web applications for a heterogeneous user group, this approach of defining static relevance weights that are valid for all users is problematic. Therefore an adaptive system requires appropriate learning mechanisms. We approach this problem by automatically adjusting the weight of a context relation for an individual user according to his interaction with the system and thus deviate from the weight defined in the context model (see Section 4).

4 Adaptation by spreading activation

In our system, activation is spread within the domain model, starting with items that are related to currently activated context factors. This way we can exploit existing relations within the domain and reduce complexity for the context engineer. The concept of spreading activation traces back to [Collins and Loftus, 1975]. Their model of spreading activation networks was originally applied in the fields of psycho linguistics and semantic priming [Anderson, 1983]. Later, the idea was adopted by computer scientists: Spreading activation techniques have successfully been used in several research areas in computer science, most notably in information retrieval ([Cohen and Kjeldsen, 1987], [Crestani, 1997] and [Berger *et al.*, 2004]). The principles of spreading activation have also been used by [Pirolli and Card, 1995] in their information foraging theory. [Kovacs and Ueno, 2006] extend classical associative spreading activation networks with "link types" and "context nodes" to generate context-adaptive recommendations. An interesting approach for using spreading activation to explore transitive associations between users can be found in [Huang *et al.*, 2004], who use this strategy to avoid the sparsity problem in collaborative filtering.

Several algorithms have been developed to implement the concept of spreading activation. However, the general idea is the same: At the beginning, one or more nodes are activated. These are called *initial nodes*. From these initial nodes, activation is propagated through the network. Once triggered the so called *pulse* passes through adjacent nodes – thereby amplifying them – until a certain termination condition is met.

4.1 The branch-and-bound algorithm

Three algorithms often used for spreading activation are examined in [Huang *et al.*, 2004]: *Constrained leaky capacitor* (originally proposed by [Anderson, 1983]), *Hopfield nets* and *Branch-and-bound*. For performance reasons we chose the branch-and-bound algorithm and our implementation of this algorithm is as follows:

Initialization: Before the actual execution of spreading activation begins, the network must be initialized:

1. The weights for the links are set based on the user's individual context model. Moreover, in our approach, the network is not necessarily in a blank state when a spreading activation run starts. Therefore, initial activation levels for each node in the network are set. These are based on the resulting activation levels of the previous run.
2. The initial nodes are activated with a certain value. The activation received by the start nodes is added to their previous state. Optionally the new activation level is calculated by applying an activation function to this sum.
3. The initial nodes are inserted into a priority queue ordered by descending activation.

Execution: After initialization, the following steps are repeated until a defined termination condition is fulfilled or the priority queue is empty. The termination condition can be configured freely, but two pre-defined termination conditions are provided: (1) A maximum of *activated* nodes is reached, (2) a maximum of *processed* nodes is reached. A processed node is a node that has itself propagated activation to adjacent nodes.

1. The node with the highest weight is removed from the queue.
2. The activation of that node is passed on to all adjacent nodes, if this is not prevented by some restriction imposed on the spreading of activation. If a node j receives activation from an adjacent node i, a new activation level is computed for j.

   $$A_j(t+1) = A_j(t) + O_i(t) \times w_{ij} \times a$$

   where $A_j(t)$ is the previous activation of j, $O_i(t)$ is the output activation of i at the time t, w_{ij} is the weight of the relation between i and j and a is an attenuation factor. The output activation of a node is the activation it has received. An arbitrary function can be used to keep the values in a predefined range.
3. The adjacent nodes that have received activation are inserted into the priority queue unless they have already been marked as processed.
4. The node that passed on its activation to the neighboring nodes is marked as processed.

Our spreading activation mechanism provides several parameters that allow the context engineer to modify its behavior, depending on the desired results and the domain. In order to prevent activation from spreading through the whole network and eventually activating every single node, we make use of constrained spreading activation: Depending on the concept type, the outgoing edges, the path-distance between nodes the spreading activation process can be influenced. Details about those constraints can be found in [Cohen and Kjeldsen, 1987] and [Rocha *et al.*, 2004]. Additionally, the attenuation factor used in the input function can be configured and reverberation can be prevented. This means that a node j must not propagate activation to a node i if node j has itself been activated by node i before in the same run. Finally, our spreading activation mechanism allows the adjustment of relation type weights. A relation type weight is used for each relation for which no individual weight has been set in the initialization phase of the algorithm.

4.2 Context reasoning

When a new session starts, a user specific Spreading Activation network is being created from the various models. In its structure and semantics those networks are the same for each user. It is individualized by adding numerical values to it: Each node receives a specific activation that represents the importance of that domain item for the respective user. In addition to their semantics, the relations also receive weights for the individual user. If a relation r between two nodes i and j has a high value this means that the relation is very important for the user. For other users the same relation r may be completely irrelevant.

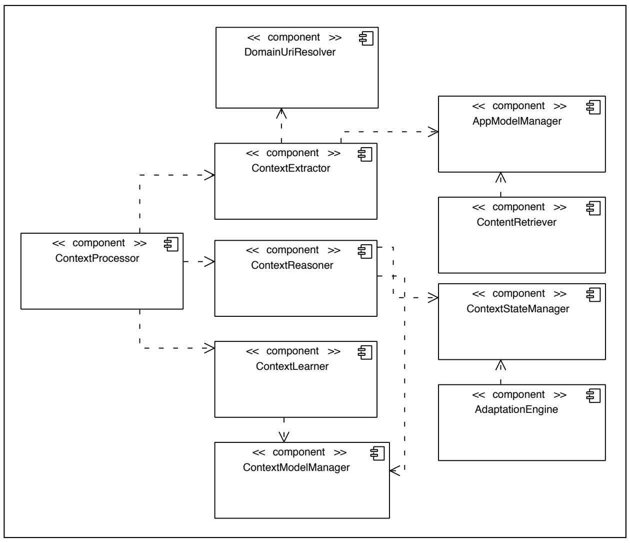
At the beginning, each node has an initial activation value of 0. Upon a request by a user, the requested content node and the sensed context factors are being used as initial nodes for the spreading activation process. As a result certain concepts and instances that seem to be important in this particular context can be used for adaptation effects. Furthermore the resulting values are being saved and can be considered for future adaptations - some of them only within the current session, some permanently. Indeed, it is neither appropriate nor does it reflect reality to let the activation rise monotonously during the entire period of usage. Because of this, we implemented a slight decay of all node weights that takes place at frequent intervals. Hence, a certain weight for a domain item can only be maintained if it is regularly activated – either directly if the user clicks on the appropriate navigation node, or indirectly by spreading activation from a related domain item.

4.3 Learning by adjusting the relation weights

As already mentioned, our system does not only manage separate node weights for each user, but also separate relation weights. A relation weight represents the importance that the relation has for an individual user. Initially, the relevance weight of a context relation is defined globally in the context model. Thus, it is the same for each user first. Adjusting the weight of a relation within the domain model is similar to the adjustment of context relation weights. When a spreading activation run is performed, each node stores the path to the initial node whose propagation led to its activation – together with information on how much activation it has received via that path. If within a certain amount of requests the user navigates to a domain item i that has previously been activated in a spreading activation run via the path p, the importance of the relations that form the path p are increased. Relation weight adjustments are stored permanently, so that the system learns what is important in a specific usage context. If the user does not "confirm" the activation path within a certain period of time by requesting the recommended node, the relation is considered to be not very important and therefore decreased in weight. This idea was inspired by Hebbian Learning [Hebb, 1949]. By doing so the spreading activation process itself becomes part of the adaptation process.

5 Architecture

Whenever the system recognizes certain context factors, activation energy is injected into the context model in order to activate relevant domain items. Based on this information, adaptations of content or navigation can be initiated. The framework focuses on context recognition, context reasoning, learning about context, and providing services for adaptation to context. The actual generation of the web pages that are sent to the browsers is not part of it, which allows for a maximum freedom concerning the technologies for generating pages. Generally, most of the main components of SPREADR can be assigned to either of two tasks: context processing or response generation. However, for some of the components, such a clear-cut classification is not possible because they are involved in both tasks. Figure 5 shows the main components as well as the dependencies between them.



<< component >> DomainUriResolver
<< component >> AppModelManager
<< component >> ContextExtractor
<< component >> ContentRetriever
<< component >> ContextProcessor
<< component >> ContextReasoner
<< component >> ContextStateManager
<< component >> ContextLearner
<< component >> AdaptationEngine
<< component >> ContextModelManager

Figure 2: Components in SPREADR

5.1 Context processing components

Context Processor

The ContextProcessor is responsible for initiating and controlling the context processing of a request. Upon each request to the server, the ContextProcessor must be provided with a HttpServletRequest object that represents the current request in order to start the context extraction, reasoning, and learning process. Therefore, it contains references to the ContextExtractor, the ContextReasoner, and the ContextLearner component (if learning of context models is desired).

Context Extractor

The first component called by the ContextProcessor upon a new request is the ContextExtractor. Its task is to extract context information from the data that is available in an HttpServletRequest object and to return the extracted context information as a collection of context factors, each of them assigned to one of the context categories described in Section 3 and has an activation level in the interval $[0, 1]$. In its default implementation, the ContextExtractor delegates its task to sub-components, each of them being responsible for extracting context factors of a single category. Once the ContextProcessor has received the extracted context factors from the ContextExtractor, it hands them over to the ContextReasoner.

Context Reasoner
This component is responsible for activating additional context factors, based on past context states and on the extracted context factors, for instance by combining context factors from several categories in order to infer additional context factors. However, although this can easily be changed by implementing an alternative ContextReasoner, the core reasoning is achieved by spreading activation in the context model. Access to the context model is provided by the ContextModelManager (see below). The spreading-activation based reasoning is performed by a corresponding sub-component. First and foremost, it is used to activate context factors representing domain items that are relevant in the current context. When the ContextReasoner has finished, it passes the context state, an artifact of its work that consists of all currently active context factors, to the ContextStateManager. Moreover, it tells the ContextModelManager to save the context model.

Context State Manager
The ContextStateManager manages past and present context states for an individual user and enables other components to access this information.

Context Learner
After context reasoning, context learning can optionally be initiated by the ContextProcessor. To do so, the latter calls the ContextLearner which accesses a context model from the ContextModelManager, modifies the model in some way and asks the ContextModelManager to save the changed model.

Context Model Manager
It is the ContextModelManager, that is responsible for managing the context model for a specific user. It provides access to the context model and, if asked to do so, persists the context model, so that potential changes that have been made to it by the Context Learner are not lost after the session has terminated. While this component is intended to provide user-specific context models, alternative implementations might provide the same model to all users. This is reasonable if no Context Learner is used.

5.2 Response generation components

While the functionality of SPREADR does not comprise the actual page generation, it provides some components that are useful for accessing the content to be rendered: the *DomainUriResolver*, the *AppModelManager*, the *ContentRetriever*, and the *AdaptationEngine*.

Domain URI Resolver
The DomainUriResolver is a utility component, that resolves the requested URI to an item or concept in the domain ontology. When using a MVC framework, a Controller can use this information to provide the view with the information that is necessary to render the requested domain item. The DomainUriResolver is also used by the ContextExtractor to extract the appropriate context factor in the context category application.

Application Model Manager
Access to the ontologies is provided by the AppModelManager.

Content Retriever
The ContentRetriever is a special component because implementations of it must be developed for the individual web application. It can then be called to get the information that is to be displayed, encapsulated in an instance of a class adhering to the JavaBean specification.

Adaptation Engine
It is the task of the AdaptationEngine to provide content and services to components that are responsible for page generation. Currently, its sole functionality is to provide items that are relevant in the current context.

6 Experiences

In order to test the effectiveness of our adaptation mechanism and to clarify our methodology, we developed an adaptive music portal. This is a typical scenario where adapting to the user and his current context is often considered to make sense. People often have a small number of favourite artists but are not aware of other artists they might like, do not notice dates of interesting concerts taking place close to their current location, or that the music they are interested in is dependent on context such as time. In our scenario, we target these problems by adapting the content of the portal to the current usage context, i.e. to the user profile enriched with activations of items by the current context. Our music portal provides album reviews, artist biographies, concert information and several kinds of additional information about events and items.

For evaluation purposes we created 4 typical usage scenarios that had to be simulated by various users. Each of the scenarios contained a part with context learning enabled (via relation weight adjustments) and a part without. The users did not know about those technical details and had to rate the quality of the adaptation effects. Those effects have been rated considerably better when context learning was enabled, because in that case they were able to find interesting items with significantly less clicks.

7 Conclusions

We introduced a novel approach to determine the most important elements of a given ontology with regard to current context. On this basis adaptation activities can automatically be performed. We call this a holistic spreading activation technique, because it relies entirely on the results of the spreading activation processes. Furthermore context relations are fully integrated into the propagation process and thereby affect the adaptation activities. Our goal was to design an adaptation mechanism for context-adaptive web applications including appropriate learning mechanisms to close the gap between context-awareness and self adaptation. [Perkowitz and Etzioni, 2000] call those web sites adaptive, which “automatically improve their organization and presentation by learning from visitor access patterns”. The proposed system meets this requirements. As for future directions, our short-term goal is to cluster user profiles and thus allow cross-network-propagation of activity action, whereas the design of truly information-centered applications with context-adaptive interfaces may be a possible long-term goal.

References

[Adomavicius and Tuzhilin, 2001] Gediminas Adomavicius and Alexander Tuzhilin. Multidimensional recommender systems: A data warehousing approach. *Lecture Notes in Computer Science*, 2232, 2001.

[Anderson, 1983] John R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 22:261–295, 1983.

[Berger *et al.*, 2004] Helmut Berger, Michael Dittenbach, and Dieter Merkl. *An Adaptive Information Retrieval System Based on Associative Networks*, volume 31 of *Conferences in Research and Practice in Information Technology*. ACS, Dunedin, New Zealand, 2004. First Asia-Pacific Conference on Conceptual Modelling (APCCM2004).

[Burke, 2002] Robin D. Burke. Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact*, 12(4):331–370, 2002.

[Chen and Kotz, 2000] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, 2000.

[Cheverst *et al.*, 2000] Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. Experiences of developing and deploying a context-aware tourist guide: The GUIDE project. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM-00)*, pages 20–31, N. Y., August 6–11 2000. ACM Press.

[Claypool *et al.*, 1999] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper, 1999.

[Cohen and Kjeldsen, 1987] Paul R. Cohen and Rick Kjeldsen. Information retrieval by constrained spreading activation in semantic networks. *Inf. Process. Manage*, 23(4):255–268, 1987.

[Collins and Loftus, 1975] Allan M. Collins and Elizabeth F. Loftus. A spreading-activation theory of semantic processing. *Psychological Review*, 82:407–425, 1975.

[Crestani, 1997] Fabio Crestani. Application of spreading activation techniques in information retrieval. *Artif. Intell. Rev*, 11(6):453–482, 1997.

[Dey *et al.*, 2004] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a CAPpella: programming by demonstration of context-aware applications. In Elizabeth Dykstra-Erickson and Manfred Tscheligi, editors, *Proceedings of the 2004 Conference on Human Factors in Computing Systems, CHI 2004, Vienna, Austria, April 24 - 29, 2004*, pages 33–40. ACM, 2004.

[Dey, 2001] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[Guarino and Giaretta, 1995] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N. J. I. Mars, editor, *Towards Very Large Knowledge Bases*, pages 25–32. IOS Press, Amsterdam, 1995.

[Hebb, 1949] Donald O. Hebb. *The Organization of Behavior*. John Wiley Sons, 1949.

[Henricksen and Indulska, 2005] Karen Henricksen and Jadwiga Indulska. Personalising context-aware applications. In Robert Meersman et al., editor, *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops, Agia Napa, Cyprus*. Springer, 2005.

[Herlocker and Konstan, 2001] Jonathan L. Herlocker and Joseph A. Konstan. Content-independent task-focused recommendation. *IEEE Internet Computing*, 5(6):40–47, 2001.

[Huang *et al.*, 2004] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems*, 22(1):116–142, 2004.

[Kaltz, 2006] Joachim Wolfgang Kaltz. *An Engineering Method for Adaptive, Context-aware Web Applications*. PhD thesis, Universitaet Duisburg-Essen, Campus Duisburg, 2006.

[Kappel *et al.*, 2003] Gerti Kappel, Birgit Pröll, Werner Retschitzegger, and Wieland Schwinger. Customisation for ubiquitous web applications a comparison of approaches. *Int. J. Web Eng. Technol*, 1(1):79–111, 2003.

[Kovacs and Ueno, 2006] Alexander I. Kovacs and Haruki Ueno. Recommending in context: A spreading activation model that is independent of the type of recommender system and its contents. In Vicent Wade, Helen Ashman, and Barry Smyth, editors, *Proceedings of the AH2006*. Springer, 2006.

[Melville *et al.*, 2002] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[Middleton *et al.*, 2004] Stuart Middleton, Nigel Shadbolt, and David De Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, 2004.

[Neches *et al.*, 1991] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):16–36, 1991.

[Oppermann, 2005] Reinhard Oppermann. From user-adaptive to context-adaptive information systems. *iCom, Zeitschrift für interaktive und kooperative Medien*, 3/2005:4–14, 2005.

[Perkowitz and Etzioni, 2000] Mike Perkowitz and Oren Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artifical Intelligence*, 118(1-2):245–275, 2000.

[Pirolli and Card, 1995] Peter Pirolli and Stuart K. Card. Information foraging in information access environments. In *CHI*, pages 51–58, 1995.

[Rocha *et al.*, 2004] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi de Aragão. A hybrid approach for searching in the semantic web. In *WWW*, pages 374–383, 2004.

[Ziegler *et al.*, 2005] Jürgen Ziegler, Steffen Lohmann, and Joachim Wolfgang Kaltz. Kontextmodellierung für adaptive webbasierte systeme. In C. Stary, editor, *Mensch & Computer 2005: Kunst und Wissenschaft*. Oldenbourg Verlag, München, 2005.