Context-aware Recommendations

Tim Hussein

University of Duisburg-Essen, Germany
http://interactivesystems.info/hussein
tim.hussein@uni-due.de

Abstract. This article illustrates the vivid research field of hybrid and context-aware recommender systems. Moreover, two own approaches to deal with context-awareness in recommender systems, are described in detail.

1 Introduction

With the help of recommendations, large collections of products, or services are made accessible. Recommender systems support users by recommending content considered as being particularly interesting for them. They play an important role in handling large amounts of information. Often, the content and artifacts a person might be interested in, depend on the specific situation: The current location, season, user role, temperature, etc. Context-aware recommender systems try to exploit the usage context to improve the recommendation generation process.

Unfortunately, to this day, no commonly accepted technical definition of context does exist. Just for the term "context", there are more than 150 definitions from various disciplines. One of the most frequently cited definitions was proposed by Abowd, Dey, and others [1]:

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user, and applications themselves."*

This rather broad definition lets the designer decide what he or she considers as relevant contextual information. This paper illustrates the state of the art regarding context-aware recommendations and introduces two techniques that incorporate both context information derived from system interaction (clickstream, history) as well as information about the external circumstances (location, time, season, etc.).

2 State of the Art

This section introduces into the research field of Recommender Systems. Beginning with basic techniques, it explains state-of-the-art approaches of combining them and, finally, integrating contextual information into a recommendation process.

2.1 Basic Recommendation Algorithms

Recommender systems have been established as an independent research area during the 1990s, having their roots in various disciplines like cognitive science and information retrieval. Most techniques can be roughly divided into content-based [2], collaboration-based [3], and hybrid approaches [4, 5].

Content-based systems incorporate features associated to the objects of interest. User ratings or transactions can be analysed in order to find out his or her interests, to recommend items similar to those bought in the past or rated as positive. For instance, neural nets, decision trees and vector-based representations can be used for that purpose. Collaborative filtering (CF) methods are supposed to be the most widely implemented recommendation techniques. They can be partitioned into classical User-based-CF and Item-based CF methods. User-based techniques identify so-called mentors for a user by generating vectors from the user's ratings and comparing those vectors, for instance, by correlation measurement or cosine. They assume that similar users are interested in similar items. So the mentors ratings for items are multiplied by the mentors similarity in order to predict the recommendations.

These computations may be time consuming and are often inappropriate for real-time recommendations with massive data sets and tens of millions of customers. Companies like Amazon.com therefore often use so-called Item-based collaborative filtering, that is based on the ideas of Sarwar et al. [6]. For each product a recommendation list of similar items is being pre-computed regularly based on what users tend to purchase together.

2.2 Hybrid Approaches

Different recommendation approaches are often combined into so called hybrid recommender systems (which may also incorporate information like social or demographic data for instance). The majority of hybrid recommender systems use collaborative filtering as the core method while content-based filtering offers solutions to the shortcomings of CF.

Balabanovic & Shoham [7] force items to be, at the same time, close to the user thematic profile, and highly rated by his neighbors as well. Pazzani [8] compares users according to their content profiles and thereupon uses collaborative filtering to generate recommendations. Other commonly cited approaches that combine content-based and collaborative filtering have been introduced by Melville et al. [9], Han & Karypis [10], and Wang et al. [11].

2.3 Context-aware Recommender Systems

Combining different techniques has been one of the main research interests within the field of recommender systems as well as incorporating contextual information [12, 13, 5, 14]. Context can be used in several ways to enhance recommender systems. Shepitsen et al. for instance introduce an agglomerative hierarchical clustering algorithm for social tagging systems [15]: A user-selected tag is used

for a context-dependent limitation of the selected set of clusters considered for the recommendation process.

A content-based model for adaptive recommendations with the help of a use context is described by Kim and Kwon [16] using a set of four ontologies from which the "use context" for a user is derived: Product, location, shopping record and consumer. Most valued products are taken from the product ontology with help of the consumer preferences and shopping record. The most valued recommendations are taken and displayed in more detail. Products are ordered in a concept hierarchy from broad to most specific. If the user chooses a concept, the context switches and more specific information is shown. In this way, context is used to control the information detail of recommendations.

Adomavicius et al. add contextual information as additional dimensions to the given user and item dimension in an collaborative filtering approach [14]. The recommendations are derived from the item ratings under the given context.

Contextual information for both the involved items and recommendation process itself is proposed by Loizou et al. [17], who use an ontology containing information about items and the recommendation process. The ontology is build with the help of web services and expanded over time with new data to match the current recommendation context at runtime. Suchlike information could for instance contain metrics for usefulness for specific users. To extract recommendations, this ontology is mapped into a vector space from which only relevant parts are sliced out.

All theses systems work well in their specific environments, but there is still a research gap regarding systematic, generic and extensible integration of context information into the recommendation process. This paper introduces two frameworks that address these questions and, beyond that, can be very useful for hybrid recommender system prototyping as different recommender techniques and context sources can smoothly be combined.

3 Experiences with SPREADR

In 1975, Collins & Loftus introduced a technique called "Spreading Activation" [18]. This model was originally applied in the fields of psycho linguistics and semantic priming [19]. 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 ([20–22]). The principles of spreading activation have also been used by Pirolli & Card [23] in their information foraging theory.

The basic concept behind Spreading Activation is that all relevant information is mapped on a graph as nodes with a certain "activation level". Relations between two concepts are represented by a link between the corresponding nodes. If for any reason one or more nodes are activated their activation level arises and the activation is spread to the adjacent nodes (and the ones related to them and so on) like water running through a river bed. Thereby the flow of activation is attenuated the more it strides away from the initially activated node(s). At the

end several nodes are activated to a certain degree that are semantically related to the concepts originally selected.

At the beginning, each node has an initial activation value of 0. We use the current context as the starting point for the Spreading Activation: When a new session starts all necessary context information is sensed and the nodes representing the recognized context factors are used as initial nodes to trigger the activation flow. As a result concepts and items that are related to the current context have a high activation value. A little example should illustrate this: The setting may be an information portal for leisure activities. Imagine the user starts a new session from Duisburg and the current time is evening. The system recognizes the time and the current location. This may result in raising the activation levels of all venues located in her city and events that take place on that particular evening are highly activated, too. Perhaps the system is configured to take local weather information into account as a context factor. So perhaps open air events may get a higher activation than indoor events. The flexible architecture allows a combination of arbitrary context information. Finally all nodes have a certain activation value that represents their degree of relevance in the current situation.

After that run the activation values are not reset but refined with every user action. If he clicks on a certain domain item - for instance a concert - this interaction is taken into account, too. The node representing that particular concert now is the initial node to another Spreading Activation run and transmit activation energy to all related concepts and items. That may include the concept of a concert in general (and thereby activating other concerts), the artist, the music genre, the venue of the concert and so on. Each interaction adopts the weights more and more to the current context and the user's interests.

At this point no adaptation is performed yet. Only the underlying models are adjusted to the current context and usage behavior. We believe that it is a good idea to separate these model adjustment mechanisms from the process of web page generation. Any changes in the page generation and presentation framework won't affect the reasoning part of the system and vice versa.

Several algorithms have been developed to implement the concept of spreading activation. Details and a comparison can be found in Huang et al. [24]. We chose the so called *Branch-and-bound* approach.

3.1 The branch-and-bound algorithm

The following steps describe a single Spreading Activation run. As mentioned earlier the activation values are not reset after each cycle so that the networks can develop into a kind of user and context profile. During a Spreading Activation run two phases can be distinguished: Initialization and execution.

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. In most cases a linear or parabolic function will be meaningful.
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.

When a new spreading activation run is triggered the values are not reset, so that the network is refined every time the process is executed. We implemented an aging mechanism by attenuating each activation value by 5% before each new spreading activation run.

3.2 Configuration

Certain constraints and termination conditions can be defined to to prevent activation from spreading through the whole network and eventually activating every single node. Additionally this allows for refining the Spreading Activation process regarding performance. The process can be influenced for instance depending on the concept type, the outgoing edges or the path-distance between nodes. Details about those constraints can be found in [20] and [25].

In addition, the sub-functions described above can be configured - for instance the attenuation factor - or reverberation can either be allowed or 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.

Details about SPREADR can be found in [26].

4 Experiences with DISCOVR

Like many other context-aware approaches (including some of the systems presented in Section 2), DISCOVR utilizes semantic background information represented as ontologies. All item- and context-related information to be incorporated by the system has to be modeled in an ontology[1], for instance *[CD Garth Brooks - Beyond the Season]* → *[suitable for holiday]* → *[Christmas]*. Surely, this is a laborious task, but DISCOVR is mainly a framework for prototypical development and the scenarios will more likely be several hundred items and concepts rather than millions.

The components introduced in the upcoming section exploit this information and build context-aware recommendations upon.

4.1 A service architecture for recommendations

One important idea that was present throughout all our concepts was to realize the different components in the form of separate *services*. Each service has some kind of input and some kind of output, so that they can be connected to each other. While each service can present its data as a semantic model, for instance for visualization and explanation purposes, most of the communication between services takes place "on rails" in the way that the output type of one service matches the input type of another. We divided the input types (which we call *triggers* because they trigger the receiving component) and the output types into five distinct categories (Figure 1):

- *Semantic models*,
- *Sorted lists of resources* from our domain model, with the sorting order based upon the importance of the given resource,
- Lists of *weighted resources* with assigned weights between 0 (unimportant) and 1 (very important),
- *Numeric values*,
- *Textual data*.

[1] It may be useful to split the information into several ontologies for later reuse of parts in other scenarios

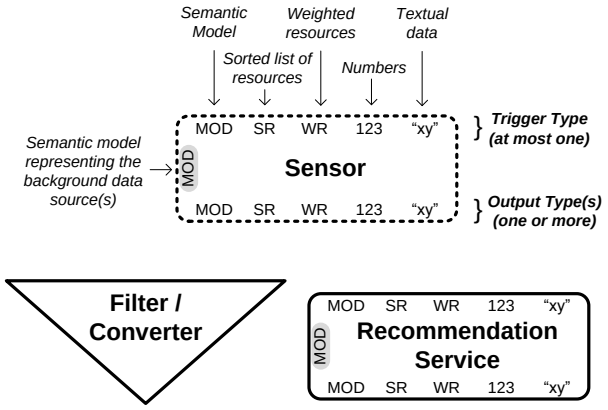

Fig. 1. Types of components of our service architecture, together with possible input and output data types.

Each service accepts triggering input of one specific type at most, which may be provided by one or more triggering services. The output of a service, on the other hand, can be manifold, depending on the particular implementation.[2] Additionally, each service can possess one or more background data source(s) in the form of semantic models (RDF triples) if needed.

In DISCOVR, we distinguish three different categories of services (Figure 1): *Sensor*, *recommendation* and *utility services*, all of them now explained in more detail.

Sensor Services A sensor is a service that acquires information about the user's context: Information derived from system interaction or information about external circumstances like the current temperature, which might be determined by contacting a web service. For a sensor to operate, it might first be necessary to retrieve already known information from another sensor: A weather sensor might need to know the user's current location, for example.

Most sensors' output has the form of a list of weighted resources (for instance closest cities along with their degree of adjacency). Some sensors, on the other hand, simply return literal values like strings or numbers (IP address, geographic coordinates).

Recommendation Services Recommendation services use their input in order to generate recommendations (context-aware ones, if the output of sensors is being incorporated). Typically, recommendation services require a background

[2] Item-based recommendations for instance could be delivered both as ordered or weighted resources.

data source containing items to recommended and their relations to each other in order to work properly.

The output of recommendation services is usually twofold: First, a recommendation service produces a list of ordered or weighted resources that can (after a possible filtering step) be presented to the user. Second, a recommendation service also returns a semantic model that is a sub-model of the background data source input and can be utilized as the background data source of yet another recommendation service, such that these services can be chained.

Utility Services The last kind of services are utility services. Such a service might *filter* its input according to specific criteria, like selecting only resources that have a specific type, or limit the amount of results. Another example of a utility service is a weighting service that assigns weights to a given, sorted list of resources according to a specific formula (virtually acting as a *converter* between sorted and weighted resources).

4.2 Experiences

We developed a virtual shopping and leisure portal including about 500 items like DVDs, CDs, sport events, concerts, etc. to demonstrate DISCOVR's potential (Figure 2) and implemented a set of components as reusable building blocks for hybrid, context-aware recommender systems.

4.3 Portal architecture

We implemented DISCOVR on top of the Spring Framework[3] as a Java web application including interfaces and abstract classes reflecting the components introduced in Section 4. In addition, we realized several concrete sensor, recommender and filter modules for testing purposes: 8 Sensors for location, clickstream, weather, season, etc.; 3 recommenders (Item-based Collaborative Filtering, Spreading Activation and Rule-based) as well as 5 filters like a Sorted Resource Weighter or a Weighted Resource Filter.

We assume that the general principles of item-based CF and rule-based recommendations are commonly known among the readership. Spreading Activation is a concept proposed in the 1970s by Collins and Loftus [18] and was originally applied in the fields of psycholinguistics and semantic priming. Later, computer scientists adopted the idea: The principles have successfully been used in several research areas in computer science, most notably in information retrieval [21] or for predicting user behavior [27]. The basic idea is that within a semantic network, certain elements are initially activated and spread this activation to adjacent elements. This activation flow runs through the network until a certain stop condition is met. In our case, the semantic model supplied by the background data source is converted into a directed graph, and those elements

[3] http://www.springsource.org

Fig. 2. Screenshot of a web-portal powered by DISCOVR including different areas for recommendations. In this case: (1) is a 'traditional' recommendation block for product recommendations based on a Spreading Activation algorithm and the latest user clicks. (2) shows local events suitable to current weather conditions. In (3) recommendations for a chosen category (e. g. CDs) are presented with certain items (with special relevance to upcoming holidays) presented as highlights (4).

reflecting the input values are initially activated and spread this energy in a highly customizable fashion within the network. The service's output is then a list of resources together with their activation weights corresponding to the activation values obtained during the activation process.

The portal makes strong use of the *MVC* design pattern, so that the information contained in the model can be displayed in various ways with the components only being loosely coupled.

Recommendations are generated using DISCOVR's service framework. A *service processor* builds a dependency graph of all registered services and then executes them in order. These services are realized as Java beans that are set up using an XML configuration file. Possible configuration parameters of a given service include the services that trigger this service, the background data source (which is, in most cases, the domain model), or service-specific items like certain filter criteria for a filtering service. A virtual service editor is planned as a future extension to make the whole setup of the service chain configurable at run-time.

4.4 Context-aware recommendations

Now we want to give an example of a service processes that produce context-aware recommendations of products associated with upcoming holidays (for instance 'Christmas', see Figure 3). For example, when the user's last click was

on a romantic movie, we would recommend (romantic) movies that are, in some way, related to Christmas.

To achieve this, the output of a Holiday Sensor triggers a rule applying service restricting the domain model to only those elements related to just this holiday. This sub-model then serves as the background data source for an item-based collaborative filtering triggered by the user's last visited item. For the sake of brevity, the additional background data required for the item-based collaborative filtering approach (clicked items of other users) is not modeled here.

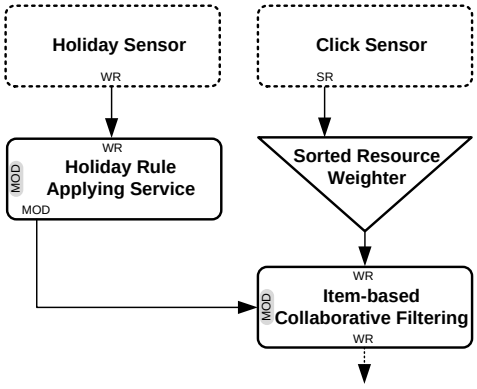

Fig. 3. Another example of a service process. In this case, a combination of a rule-applying selection service and an item-based collaborative filtering approach has been chosen.

[28] contains details on the DISCOVR Framework.

5 Summary

This article introduced the field of context-aware (and hybrid) recommendation generation. Furthermore, two approaches have been explained in detail.

6 Acknowledgements

The research presented in this paper is part of the CONTICI project, in which the Universities of Duisburg-Essen, Siegen, Hagen, and Aachen take part. CONTICI is funded by the German Research Foundation (Deutsche Forschungsgemeinschaft).

References

1. Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness.

In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer.

2. Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204, New York, NY, USA, 2000. ACM.
3. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.
4. Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*. ACM, 1999.
5. Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
6. Badrul Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Item-based collaborative filtering recommendation algorithms. In Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko, editors, *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, Hong Kong, 2001. ACM.
7. Marko Balabanovic and Yoav Shoham. Combining content-based and collaborative recommendation. *Communications of the ACM*, 40:66–72, 1997.
8. Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.
9. Prem Melville, Raymond 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.
10. Eui-Hong Han and George Karypis. Feature-based recommendation system. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 446–452, New York, NY, USA, 2005. ACM.
11. Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA, 2006. ACM.
12. Jonathan L. Herlocker and Joseph A. Konstan. Content-independent task-focused recommendation. *IEEE Internet Computing*, 5(6):40–47, 2001.
13. Gediminas Adomavicius and Alexander Tuzhilin. Multidimensional recommender systems: A data warehousing approach. In *WELCOM '01: Proceedings of the Second International Workshop on Electronic Commerce*, pages 180–192, London, UK, 2001. Springer.
14. Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.
15. Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender Systems (RecSys)*, pages 259–266, New York, NY, USA, 2008. ACM.

16. Sungrim Kim and Joonhee Kwon. Effective context-aware recommendation on the semantic web. *International Journal of Computer Science and Network Security*, 7(8):154–159, 2007.
17. Antonis Loizou and Srinandan Dasmahapatra. Recommender systems for the semantic web. In *ECAI 2006 Recommender Systems Workshop*, 2006.
18. Allan M. Collins and Elizabeth F. Loftus. A spreading activation theory of semantic processing. *Psychological Review*, 82(6):407–428, 1975.
19. John R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 22:261–295, 1983.
20. Paul R. Cohen and Rick Kjeldsen. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management*, 23(4):255–268, 1987.
21. Fabio Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11(6):453–482, 1997.
22. Helmut Berger, Michael Dittenbach, and Dieter Merkl. An adaptive information retrieval system based on associative networks. In *APCCM '04: Proceedings of the first Asian-Pacific conference on Conceptual modelling*, pages 27–36, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.
23. Peter Pirolli and Stuart Card. Information foraging in information access environments. In Irvin R. Katz, Robert Mack, Linn Marks, Mary Beth Rosson, and Jakob Nielsen, editors, *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 51–58, Denver, Colorado, USA, 1995. ACM Press.
24. 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.
25. Cristiano Rocha, Daniel Schwabe, and Marcus Poggi Aragao. A hybrid approach for searching in the semantic web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 374–383, New York, NY, USA, 2004. ACM.
26. Tim Hussein and Jürgen Ziegler. Adapting web sites by spreading activation in ontologies. In Lawrence Bergman, Kim, Jihie, Bamshad Mobasher, Stefan Rueger, Stefan Siersdorfer, Sergej Sizov, and Markus Stolze, editors, *Proceedings of International Workshop on Recommendation and Collaboration*, New York, USA, 2008. ACM.
27. Wai-Tat Fu and Peter Pirolli. Snif-act: a cognitive model of user navigation on the world wide web. *Human-Computer Interaction*, 22(4):355–412, 2007.
28. Tim Hussein, Timm Linder, Werner Gaulke, and Jürgen Ziegler. Context-aware recommendations on rails. In *Workshop on Context-Aware Recommender Systems (CARS-2009) in conjunction with the 3rd ACM Conference on Recommender Systems (ACM RecSys 2009)*, New York, NY, USA, 2009.