A heuristic as basis for an adaptive e-commerce recommender system

Elmar P. Wach
STI Innsbruck
University of Innsbruck / Elmar/P/Wach eCommerce Consulting
6020 Innsbruck, Austria / 22339 Hamburg, Germany
elmar.wach@sti2.at / wach@elmarpwach.com

***Abstract*—The research described in this paper proposes an evolution heuristic for realising an adaptive semantic e-commerce recommender system by establishing a feedback cycle. This recommender extracts questions from product domain ontologies (PDO) which are used in the dialogue of the recommendation process. The heuristic decides an automated PDO evolution (without a human inspection) in order to realise an automatic adaptation of the recommendation process. The feedback is derived from user interactions with the user interface of the recommender. This research shows that the automated PDO evolution outperforms a manual one. The evolution heuristic has been evaluated with an experiment and validated in real-world testing series.**

***Keywords*-ontology evolution; heuristics; recommender systems; self-adapting information systems**

I. Introduction

Recommender systems in e-commerce applications have become business relevant in filtering the vast information available in e-shops (and the Internet) to present useful product recommendations to the user. The recommendation process of such systems is static[1] most of times, even though the 'world' is dynamic. New products are introduced in the market whereas old ones vanish. Hence, the products offered in a Web application change[2], and the recommendations have to be based on the currently offered range of goods. Also, new products probably have other product features—this is especially valid for consumer goods and electronic devices. Those new features can be included in the evaluation of the recommendation process—or in a dialogue-based approach reflected in the questions to the customer—which finally leads to the recommendations.

All these reasons compel to adapt the recommendation process in order to provide useful and successful recommendations to the customer. A manual change of this process, however, is usually inefficient and, moreover, very expensive. Additionally, it is very difficult for humans to predict improvements for given changes of the recommendation process[3]. The ideal is to automatically adapt the recommendation process in order to be more efficient and to minimise the costs entailed by a manual effort.

[1]I.e., there is no automated change of the recommendation process
[2]There may be several reasons for changing the offered products
[3]Of course, this is also difficult for machines

II. Preliminaries

The research described in this paper uses a real-world conversational content-based e-recommender system—named SMARTASSISTANT—which is based on given product domain ontologies (PDO). This system has been described according to the 'Adaptation Strategy' formulated by the author [1]. With the Adaptation Strategy the ontology strategies and the ontology heuristic—which is addressed in this paper—have been defined as well. Moreover, the result of applying the Adaptation Strategy provided the basis for the technical specification and thus the programming of the new 'Adaptation Layer'. The description of this implementation is omitted due to space limitations.

The PDO[4] describe the products offered in the e-commerce application formally—which is in OWL[5] and according to GoodRelations[6]—and thus offer a higher computability than conventional product descriptions which facilitates the automated processing of information. SMARTASSISTANT extracts questions from the PDO about the product characteristics and features to investigate the user preference and eventually recommend products that match the needs of the user. By changing the PDO, this system generates different questions and/or their order and herewith adapts the recommender interface to the user preference. Hence, an automated adaptation of the recommendation process can be realised by automatically changing the PDO which also minimises the high cost of the manual adaptation of the recommendation process as well as of the underlying PDO.

This is accomplished by implementing and utilising a feedback cycle. The user feedback is gathered by unobtrusively monitoring customer needs[7] (i.e., implicit feedback). The interaction of the customer with the recommender is analysed. The key metric to measure its success is the share of customers who follows a recommendation. This metric is defined as click-out rate 'COR' (i.e., clicks-to-recommendations) which is fed back. The COR has been the basis for the PDO adaptations up to date. In this re-

[4]The description of the PDO is omitted due to space limitations
[5]Web Ontology Language, www.w3.org/TR/owl-ref
[6]An upper ontology, www.purl.org/goodrelations
[7]This part is out of scope of this research

978-1-4673-5119-5/12/$31.00

search, however, the impact of the evolution in the precedent feedback cycle is evaluated by transforming the COR to the Success Trend *ST*[8]. A positive *ST* is a positive trend (i.e., an increase) of the COR, a negative the opposite. The larger the figure is, the stronger the development of the COR (in either direction) from the precedent to the current cycle has been. Summarised, higher *ST* values are better than lower values, and a negative *ST* indicates a worse COR.

The PDO gets evolved according to the evolution strategy and evolution heuristic, and the instances[9] are newly annotated to the changed PDO if necessary. The recommendation process is successful if the customer clicks on the recommended product (i.e., clicks out). It is not successful if the customer quits the recommendation process. The COR is measured again which concludes the feedback cycle.

This research proposes predefined Types of PDO Changes according to the user dialogue in the user interface of the e-commerce recommender system. Unless the PDO changes have an impact on the user interaction, no effective feedback cycle can be established. Discussions with the application provider[10] clarified the Types of PDO Changes and their effects in the user interface. Derived from that, three Types of PDO Changes have been defined which are described in the following including their impact on the user dialogue. Figure 1 illustrates the relevant parts of the user interface for recommending digital cameras (in German) and correlates it with the entities of the respective PDO.

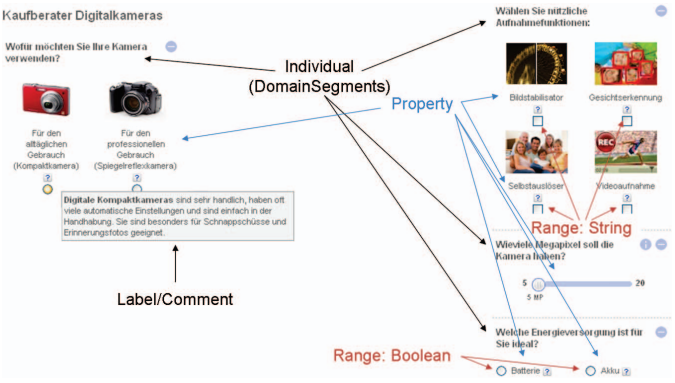

Figure 1. User interface and entities of the PDO

'Switch annotation properties label and comment': Each property is annotated with labels and comments which describe it. Those are switched, though, most of times, this makes sense when the labels and comments of all properties are switched at once. The most obvious case is switching the labels and comments between different languages. A more specific case is switching the labels and comments to ones which are extracted from a different information source like labels and comments for a general audience to labels and comments for a technophile audience. The impact on the user dialogue is a textual modification of the questions and—referred to the more specific case from above—a different sales approach. It would be need-based (i.e., the labels and comments are based on the source for a general audience) or technology-prone (i.e., the labels and comments are based on the source for a technophile audience).

'Switch datatype property ranges': A datatype property has as range a literal which is exchanged. A group of properties can be switched from a Boolean range to a string range and vice versa. This only makes sense, in case each property of an individual has the same range. The impact on the user dialogue are different options for selecting the properties. For example, in figure 1 the individual 'Recording'(i.e., 'Aufnahmefunktionen' in the upper right corner) contains only properties with the range string, namely 'pictureStab' (i.e., 'Bildstabilisator'), 'faceIdentification' (i.e., 'Gesichtserkennung'), 'selftimerFunction' (i.e., 'Selbstausloser'), and 'videofunction' (i.e., 'Videoaufnahme'). In case the range of these properties are switched to Boolean, they would be listed in the user interface with a radio button (like 'Akku' in the lower right corner in figure 1) instead of a check box.

'Switch individuals': An individual (e.g., WeightAndSize) groups specific properties of the PDO (e.g., height, length, weight, width) in DomainSegments. Such a group of properties is switched to another individual (e.g., GeneralCharacteristics). The impact on the user dialogue is a different clustering of the questions. The questions regarding the respective group of properties would be displayed in another area of the user interface.

III. Related Work

In Computer Science changing an ontology is referred to 'ontology evolution'. In that research area has been put a lot of effort. Nevertheless, an automated ontology evolution (without a human inspection) has not completely been realised to the best knowledge of the author. Additionally, the research described in this paper is also a novel approach because it proposes an automated PDO evolution induced by the automated processing of user feedback. This section introduces existing ontology evolution algorithms.

[2] adopted and compared two approaches from the database community. The procedural approach is based on consistency constraints and rules to be followed to maintain those constraints. The declarative approach is based on a sound and complete set of axioms that formalises the dynamics of the evolution. The ontology engineer expresses the request for a change in a declarative manner, and the change gets processed automatically.

[3] considered an ontology as a logical theory that can only become inconsistent by adding axioms.[11] Consistency

[8]This transformation is omitted due to space limitations

[9]The instances of a PDO are the respective products and their descriptions

[10]Smart Information Systems GmbH

[11]Due to the monotonicity of the description logic/OWL DL

is ensured by determining a minimal part of the ontology (i.e., a set of conflicting axioms) to be removed from the ontology when an axiom was added. Those axioms are considered to be structurally connected with the conflicting axioms. The presented algorithm finds the maximal consistent sub-ontology by removing and adding the respective axioms (similar to belief contradiction and belief revision operations in the theory of belief change, for example, the AGM theory [4]).

According to [5], an ontology based on Description Logic (like OWL DL) can contain contradictions only if its underlying logic allows negation. The authors proposed the introduction of a new concept, in case of a conflicting axiom according to a confidence rating (i.e., structural connectedness) computed with the term distribution in a text document to be learned. The authors present two algorithms which can be processed successively—AdaptOnto adapts a TBox[12] to a new axiom by removing concept descriptions, and Regen resolves under-generalised concepts by inducing new concept definitions (i.e., generalise) in the TBox.

[6] and [7] exploited the research in the field of belief change which is 'fact-centered' and transferred it to ontology evolution which is mainly treated as 'modification-centered' (i.e., the fact that initiates the change is not important, and the system input is just the change to be executed). They state that the representation of the knowledge affects the ontology evolution algorithm: In case the explicitly represented knowledge serves as justification for our beliefs (i.e., a belief base), it can directly be changed whereas the implicit knowledge is only affected by changes in the explicit knowledge.

[8] pursue the belief revision principle of minimal change—the resulting ontology should be as 'close' as possible to the original one. The ontology model is instantiated with RDF[13] mapped to First Order Logic predicates. These predicates are ordered according to preference allowing an evolution decision based on the belief revision principle of minimal change. An evolution algorithm for arbitrary changes rather than a predetermined set of changes has been developed. It processes all predicates and determines possible ways for solving an inconsistency.

Due to the specific challenges for the research described in this paper like creating an automated ontology evolution based on user feedback and utilising predefined PDO changes, none of the discussed algorithms can be reused. Hence, a specific PDO evolution algorithm would have to be formulated which would be application-specific. The effort for formulating an adequate algorithm is too high compared with the pragmatic approach of predefining types of PDO changes which also ensure a consistent ontology evolution. Moreover, such a specific algorithm could be reused neither in other research efforts nor in other applications.

[12]The TBox contains the axioms of an ontology

[13]Resource Description Framework, www.w3.org/TR/rdf-concepts

IV. Evolution Heuristic

Instead of formulating such an ontology evolution algorithm, the challenge of an automated PDO evolution is solved with generic evolution strategies and an application-specific evolution heuristic. The evolution strategy—which is manually selected in the Administration Interface of the Adaptation Layer—decides the general evolution behaviour (e.g., executing the same Type of PDO Change or a rollback) by correlating the Types of PDO Changes needed with the *ST* calculated. Four evolution strategies have been defined. The description of those are omitted due to space limitations. The interested reader is referred to [1].

A heuristic is a strategy that uses accessible and loosely applicable information to solve a problem of a human being or a machine [9] and leads to a solution of a complex problem with simplified conceptual aspects or reduced computation power. [10] mentioned first the term metaheuristic for a computational method that makes few or no assumptions about the problem being optimised and introduced the tabu search metaheuristic [11]. The tabu search enhances a local search metaheuristic (i.e., iteratively improving a criterion in the search space) by using 'taboos'—a solution is not executed again according to criteria defined in the tabu list.

This research focuses on the tabu search metaheuristic. The application-specific evolution heuristic eventually determines the concrete PDO change to be executed (e.g., switching the group of properties with 'weight' from the individual WeightAndSize to the individual GeneralCharacteristics).

The general characteristics of the evolution heuristic are concisely described in the following. Firstly, always the impact of the evolution in the precedent feedback cycle is evaluated. Secondly, only one implicit PDO change (i.e., a PDO change of the Type of PDO Change 'Switch datatype property ranges', 'Switch individuals', or 'Switch annotation properties label and comment') is executed per feedback cycle. Thirdly, each PDO change which satisfies a tabu is added to the tabu list. Finally, a PDO change can not only be executed but also be reverted at one point of time. For this purpose there are two types of *ST* for determining the PDO change to be executed, namely ST_f (or *ST* Forward) which indicates the *ST* for the forward PDO change and ST_b (or *ST* Backward) which indicates the *ST* for the backward PDO change (i.e., reverts the forward PDO change). The evolution heuristic has a 'greedy' approach—it selects the PDO change with the highest precedent *ST* which has no tabu and according to the evolution strategy.

For the evolution heuristic (i) general and (ii) specific tabu criteria are defined which restrict the number of similar PDO changes. (i) avoids a uniform optimisation and cycles. For this, the PDO changes of the same Type of PDO Change are consecutively executed only as often as there are implicit Types *T* of PDO Changes (e.g., the use case has three

implicit Types of PDO Changes, hence, T = 3). In case a Type of PDO Change has less than T PDO changes, the general tabu criterion is satisfied when all PDO changes of this respective Type of PDO Change have been executed.

(ii) is specifically calculated for each Type of PDO Change. The tabu is defined as 'Type of Change' *ToC* which states the number of consecutive PDO changes of the same Type of PDO Change. *ToC* is calculated with the values for the two tabus 'set switches' s_{sw} and 'set iterations' s_{it}.

s_{sw} is the number of consecutive switches of the same Type of PDO Change (in one direction) of one set s of an ontological entity of a PDO to a switch candidate *cand*. For example, a switch of one set of property to another individual. Often, there is only one PDO change possible like changing the individual, the property range, or the annotation properties label and comment, and the next change would revert that change. This tabu is defined as follows:

$$s_{sw} = \begin{cases} cand, \text{case: } s \text{ is not related to a switch candidate} \\ cand - 1, \text{case: } s \text{ is related to a switch candidate} \end{cases} \quad (1)$$

where *cand* is the number of switch candidates of a Type of PDO Change (i.e., to these candidates can be switched), s is one set of an ontological entity of a Type of PDO Change (e.g., specific properties).

Result is the number of consecutive set switches s_{sw} to satisfy this tabu. In case s is already related to a switch candidate, s_{sw} is lessen by this one 'impossible' switch. In case s_{sw} is satisfied, another set of an ontological entity of the same Type of PDO Change is switched (case: $cand \leq T$) or another Type of PDO Change is executed (case: $cand > T$).

s_{it} is the number of consecutive switches of the same Type of PDO Change (in one direction) of the different sets s of an ontological entity of a PDO (the premise for switching another set than in the precedent feedback cycle is a satisfied s_{sw}). This tabu is defined as follows:

$$s_{it} = \frac{(S - s_{cand})}{n}, \text{case: } s \in S, n \in \mathbb{N}\backslash\{0\}, s_{cand} \subseteq S \quad (2)$$

where S are all sets of an ontological entity of a Type of PDO Change (e.g., all sets of properties, all sets of annotation properties label and comment), s is one set of an ontological entity of a Type of PDO Change (e.g., specific properties), s_{cand} is the number of sets of an ontological entity of a Type of PDO Change already related to a switch candidate, n is the fraction of the sets which are not related to a switch candidate (e.g., n = 1: All sets of those entities are allowed to be switched, n = 2: Half of those sets are allowed to be switched, etc.; n can be changed for each Type of PDO Change in the Administration Interface of the Adaptation Layer).

Result is the number of consecutive set iterations s_{it}[14] to satisfy this tabu.

The tabu Type of Change *ToC* is calculated with the overall number of consecutive PDO changes s_{sw} and s_{it} where s_{sw} respects the sets already related to a switch candidate. Additionally, it is distinguished if single sets s (i.e., $s \in S$) or all sets S at once (i.e., $s \equiv S$) are switched. This tabu is defined as follows:

$$ToC = \begin{cases} s_{sw} * s_{it} + (cand - 1) * s_{cand}, \text{case: } s \in S \\ cand, \text{case: } s \equiv S \end{cases} \quad (3)$$

Result is the number of consecutive PDO changes *ToC* of the same Type of PDO Change to satisfy this tabu. It is satisfied when having executed either all s_{sw} and s_{it} switches of the respective Type of PDO Change (case: $ToC \leq T$) or T times the number of switches of the respective Type of PDO Change (case: $ToC > T$). In case *ToC* or T is satisfied (in the use case this would be when the same Type of PDO Change shall be consecutively executed for the fourth time), another Type of PDO Change is executed, and the overall oldest tabu is deleted from the tabu list.

The tabu calculation is illustrated with the following example for the Type of PDO Change 'switch individuals' which is depicted in figure 2. Six properties of the PDO (i.e., product features; on the left side in the figure) are related to six individuals (on the right side). Usually, several properties are related to one individual. In the beginning, no forward PDO change has been executed.

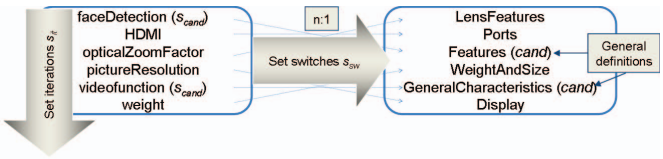

Figure 2. Tabu calculation example

For calculating the tabu 'set switches' s_{sw}, the number of switch candidates *cand* of a Type of PDO Change (i.e., to these candidates can be switched) have to be determined. It is obvious that it does not make sense to switch a property to any individual (e.g., 'faceDetection' is not a port). Hence, only individuals which are general definitions are switch candidates. In the example these are the individuals 'Features' and 'GeneralCharacteristics'. With equation 1 s_{sw} equals 2 respectively 1 for the properties 'faceDetection' and 'videofunction' which are already related to one of the two switch candidates. After two consecutive switches of the same set of properties, another set of properties is switched

[14] s_{it} is truncated to the natural number

to a general individual (the properties 'faceDetection' and 'videofunction' are allowed to be switched only once).

For calculating the tabu 'set iterations' s_{it}, *S* is determined to be 6 because there are six sets of properties. s_{cand} equals 2, namely the properties 'faceDetection' and 'videofunction'. *n* is exemplarily set to be 2 (i.e., half of the sets which are not related to a switch candidate are allowed to be switched). It does not make sense to switch all properties to one individual because then all questions regarding the properties would be displayed in one/the same area of the user interface. That would be neither user-friendly nor sales-effective. With equation 2 s_{it} equals 2.

Eventually, the tabu Type of Change *ToC* is calculated with equation 3 to be 6 for the case that single sets of the properties are switched. As the general tabu criterion *T* equals 3 (i.e., the use case has three implicit Types of PDO Changes), another Type of PDO Change is executed when 'switch individuals' shall be consecutively executed for the fourth time.

The description of the ramp-up of the evolution heuristic is omitted due to space limitations.

V. Evaluation and Validation

The evolution heuristic is evaluated by conducting an experiment. The e-commerce recommender system is validated by conducting testing series.

A. Experiment

This section shows the results of comparing the automated PDO evolution with an evolution manually decided by the ontology experts based on the delivered user feedback.

The experiment was an online survey and consisted of fifty questions about the evolution heuristic. The concrete PDO change to be executed in the upcoming feedback cycle (i.e., one per question) should be selected in order to achieve the 'best' PDO evolution from the tester point of view. The experiment was conducted with twenty ontology experts. Attendants were students of a Semantic Web course at the University of Innsbruck as well as randomly selected ontology experts who joined over the Internet. The answers where stored in the back office of the online survey application.

The experiment is evaluated by calculating the recall for the set *S* of PDO changes which has been selected by the participants of the experiment. Each of the twenty ontology experts who participated in the online survey answered fifty questions. Hence, 1,000 PDO changes were evaluated by the participants of the experiment, and *S* equals 1,000.

$$recall(S) := \sum(|apc_i = s_i|)/|S|, apc, s, S \in PC \qquad (4)$$

where *apc* is an automated PDO change, *s* is a PDO change selected by the participants of the experiment, *i* is the index for an answer to a question, *S* is the set of PDO changes evaluated by the participants of the experiment, and *PC* is the set of PDO changes *pc*. The higher the recall the better is the automated PDO evolution.

The overall result is that 833 PDO changes were selected by the human experts like the evolution heuristic did. The *recall(S)* for the set *S* of PDO changes is calculated with equation 4 to be 83.3%. Hence, more than four out of five PDO changes selected by the evolution heuristic were identical to the ones selected by the human experts.

The drill-down analysis to the question level shows that the maximum *recall(S)* is 100%. It was achieved with two questions. Hence, those questions were selected completely identical by the evolution heuristic and the participants. On the other hand, the minimum *recall(S)* is 65% (i.e., 13 participants out of twenty answered 'correctly') which was achieved with one question. This low figure could be caused by design issues in the online survey combined with similar figures shown in the table which contained all PDO changes and achieved *ST*.

The most interesting result is that some experts did not revert the PDO changes which they had selected even though those changes did not generate a good *ST*. The backward PDO changes had recalls ranging only between 70% and 80%. It seems that those participants had difficulties in accepting that their decision did not lead to a successful impact on the recommendation dialogue with the user.

This leads to the conclusion that the evolution heuristic overcomes this kind of 'human weakness' and reverts a PDO change whenever the *ST* Backward is higher.

The overall result of the experiment is that the automated PDO evolution almost always leads to the same PDO changes as the manual PDO evolution.

B. Testing Series

This section delivers the results of measuring the effects of the Adaptation Layer in a real-world scenario. Basically, the impact of changes in the user interface on the customer interaction and on the success of the SMARTASSISTANT is evaluated. As this system was applied in live e-shops of large Austrian and German online retailers, it has been a real-world scenario. The customers were unaware of the test.

The validation scenario was to analyse and evaluate the impact of the PDO evolution in the domains Blurayplayer and Camcorder after having accomplished a minimum number of 400 recommendation processes. The test row ran from December 19, 2011 to January 2, 2012. The recommendation processes where evaluated with an A/B test in which the users of the recommender were equally distributed to either variant. One variant was based on a manually evolved PDO, the other on an automatically evolved one.

In the domain of Blurayplayer the manually evolved PDO was tested in 806 recommendation processes which led to a COR of 0.458, whereas the automatically evolved PDO was tested in 825 recommendation processes and led to COR of 0.509. This is a significant increase of 11.14%.

The automated PDO evolution led to an increased success. The purple line in figure 3 shows the COR of the manually evolved PDO, the khaki line of the automatically evolved PDO.

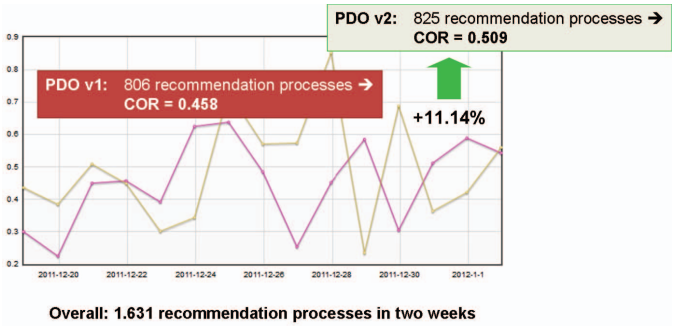

Figure 3. Graphs for the click-out rates COR of Blurayplayer

In the domain of Camcorder the manually evolved PDO was tested in 205 recommendation processes and led to a COR of 0.444. The automatically evolved PDO was tested in 233 recommendation processes and led to a COR of 0.485. This is an increase of 9.23%. The automated evolution led to an increased success for this domain as well.

The click-out rates generated with the evolution heuristic and evolution strategy are clearly higher than the ones generated with a manual PDO evolution and lead to a significantly increased success most of times. The increases of 11.14% and 9.23% respectively are impressive. In addition to that, the overall creation of the recommendation dialogue was feasible with a decreased manual work by up to 60%, according to the producer of the recommender[15].

VI. Conclusion

Automatically processing user feedback and creating an automated ontology evolution are relevant topics especially in feedback-based domains and applications like in the e-commerce industry. The better the customer is understood the better the recommendations in the application can be customised. In turn, she will 'reward' a better service level with more purchases.

By programmatically implementing the relevant characteristics of the heuristic as well as calculating the relevant metrics, a complete automation of the decision process for the PDO evolution has been accomplished. With the process described, the PDO adapts to the user feedback automatically without any explicit, direct intervention from a human. Additionally, the results show an increased success of the adaptive e-commerce recommender system.

Acknowledgment

The research presented in this paper is funded by the Austrian Research Promotion Agency (FFG) and the Federal Ministry of Transport, Innovation, and Technology (BMVIT) under the FIT-IT "Semantic Systems" program (contract number *825061*).

[15]Smart Information Systems GmbH

References

[1] E. P. Wach, "Automated ontology evolution for an e-commerce recommender," in *BIS (Workshops)*, ser. Lecture Notes in Business Information Processing, W. Abramowicz, L. A. Maciaszek, and K. Wecel, Eds., vol. 97. Springer, 2011, pp. 166–177. [Online]. Available: http://dblp.uni-trier.de/db/conf/bis/bisw2011.html#Wach11

[2] L. Stojanovic, "Methods and tools for ontology evolution," Ph.D. dissertation, Karlsruhe Institute of Technology, 2004, http://d-nb.info/1001606787. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000003270

[3] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure, "A framework for handling inconsistency in changing ontologies," in *Proceedings of the 4th international conference on The Semantic Web*, ser. ISWC'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 353–367. [Online]. Available: http://dx.doi.org/10.1007/11574620_27

[4] C. E. Alchourròn, P. Gärdenfors, and D. Makinson, "On the logic of theory change: Partial meet contraction and revision functions," *Journal of Symbolic Logic*, vol. 50, pp. 510–530, 1985.

[5] E. Ovchinnikova and K.-U. Kühnberger, "Aspects of automatic ontology extension: adapting and regeneralizing dynamic updates," in *Proceedings of the second Australasian workshop on Advances in ontologies - Volume 72*, ser. AOW '06. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2006, pp. 51–60. [Online]. Available: http://dl.acm.org/citation.cfm?id=1273659.1273666

[6] G. Flouris and D. Plexousakis, "Handling ontology change: Survey and proposal for a future research direction," Tech. Rep., 2005.

[7] G. Flouris, D. Plexousakis, and G. Antoniou, "Evolving ontology evolution," in *Proceedings of the 32nd conference on Current Trends in Theory and Practice of Computer Science*, ser. SOFSEM'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 14–29. [Online]. Available: http://dx.doi.org/10.1007/11611257_2

[8] G. Konstantinidis, G. Flouris, G. Antoniou, and V. Christophides, "Ontology evolution: A framework and its application to rdf," in *Proceedings of the Joint ODBIS & SWDB Workshop on Semantic Web, Ontologies, Databases (SWDB-ODBIS-07)*, 2007.

[9] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984.

[10] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, May 1986. [Online]. Available: http://dx.doi.org/10.1016/0305-0548(86)90048-1

[11] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.