Supporting Consumers in Providing Meaningful Multi-Criteria Judgments

Friederike Klan
Institute of Computer Science
Friedrich-Schiller-University of Jena
friederike.klan@uni-jena.de

Birgitta König-Ries
Institute of Computer Science
Friedrich-Schiller-University of Jena
birgitta.koenig-ries@uni-jena.de

ABSTRACT

The huge amount of products and services that are available online, makes it difficult for consumers to identify offers which are of interest to them. Semantic retrieval techniques for Web Services address this issue, but make the unrealistic assumption that offer descriptions describe a service's capabilities correctly and that service requests reflect a consumer's actual requirements. As a consequence, they might produce inaccurate results. Alternative retrieval techniques such as collaborative filtering (CF) mitigate those problems, but perform not well in situations where consumer feedback is scarce. As a solution, we propose to combine both techniques. However, we argue that the multi-faceted nature Web Services imposes special requirements on the underlying feedback mechanism, that are only partially met by existing CF solutions. The focus of this paper is on how to elicit consumer feedback that can be effectively used in the context of Web Service retrieval and how to support users in that process. Our main contribution is an algorithm that suggests which service aspects should be judged by a consumer. The approach effectively adjusts to user's ability and willingness to provide judgments and ensures that the provided feedback is meaningful and appropriate in the context of a certain service interaction.

Categories and Subject Descriptors

H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval; H.3.5 [**Information Storage And Retrieval**]: On-line Information Services

General Terms

Algorithms, Human Factors, Measurement

Keywords

recommend what to judge, multi-criteria judgments, personalized feedback elicitation

1. INTRODUCTION

The huge amount and heterogeneity of information, products and services that are available online, makes it difficult for consumers to identify offers which are of interest to them. Hence, new techniques that support users in the product search and selection process are required. In the past decade, semantic technologies have been developed and leveraged to approach this issue [3]. They provide information with a well-defined and machine-comprehensible meaning and thus enable computers to support people in identifying relevant content. This idea is not restricted to information, but also applies to functionality provided via the web as services. Semantic Web Services (SWS) provide a specific functionality semantically described in a machine-processable way over a well-defined interface. Similarly, service requesters may semantically express their service requirements. Having both, a semantic description of a consumer's needs as well as the published semantic descriptions of available Web Services, suitable service offers can be automatically discovered by comparing (matching) the given service request with available offer descriptions. Services might be automatically configured and composed and finally invoked over the web.

Existing semantic matchmaking and service selection approaches evaluate the suitability of available service offers exclusively by comparing the published offer descriptions with a given request description. They implicitly assume that offer descriptions describe a service's capabilities correctly and that service requests reflect a consumer's actual requirements. The first assumption might have been valid in a market with a small number of well-known and accredited companies. However, it is no longer true in today's market, where easy and cheap access to the Internet and the emergence of online marketplaces that offer easy to set up online storefronts enable virtually everyone to provide his own online shop accessible to millions of buyers. The situation becomes even more critical, since due to the huge number of offers, a hard competition and price war has been aroused that might cause some providers to promise more than they are able to provide. In our mind, the assumption that service requests reflect a consumer's actual requirements is also not realistic. This is due to the fact that, though SWS approaches provide adequate means to semantically describe service needs, they require the user to do this at a formal, logic-based level that is not appropriate for the average service consumer in an e-commerce setting. As a result, SWS applications typically provide request templates for common service needs. Those templates are then adjusted to fit to a consumer's requirements in a certain purchasing situation.

Though the resulting service requests might be a good estimate of a consumer's service needs, they cannot exactly met his true requirements. As a consequence, service discovery mechanisms that are purely based on the comparison of semantic request and offer descriptions might produce inaccurate results and thus lead to suboptimal service selection decisions.

To mitigate those problems, alternative retrieval techniques such as collaborative filtering [9] have been developed. Those techniques do not rely on explicit models of consumer requirements and product properties. They evaluate product ratings of neighboring users, i.e. those that have a similar taste, to recommend products or services that might be of interest to a potential consumer. Though collaborative filtering approaches are very effective in many domains, they lack the powerful knowledge representation and matchmaking capabilities provided by SWS and thus perform not well in situations where feedback is scarce [9]. As a solution, we propose to combine both techniques. More specifically, we suggest to perform retrieval based on semantic service descriptions and then use a collaborative feedback mechanism to verify and refine those results. We think, that such a hybrid approach can benefit from the best of both worlds and thus has the potential to significantly improve the retrieval quality. Combining semantic retrieval with collaborative feedback mechanisms is not new (see for example [8, 11]). However, we argue that simply re-using existing techniques, as done in other approaches, will not tap the full potential of this type of approach. This is due to the fact, that the multi-faceted nature and the peculiarities of SWS impose special requirements on the underlying feedback mechanism and in particular on the properties of the consumer feedback that is required. In this paper, we will analyze those requirements (Sect. 2) and will show that they are only partially met by existing collaborative filtering solutions (Sect. 3). The focus of this paper is on how to elicit consumer feedback that can be effectively used in the context of SWS retrieval and how to support users in that process (Sects. 4 and 5). Our main contribution is an algorithm that suggests which service aspects should be judged by a consumer (Sect. 6). The approach accounts for a user's ability and willingness to provide judgments and ensures that the provided feedback is meaningful and appropriate in the context of a certain service interaction. Our evaluation results show that the proposed procedure effectively adjusts to a consumer's personal judgment preferences and thus provides helpful support for the process of feedback elicitation (Sect. 7). A detailed discussion on how to effectively use consumer feedback to enhance SWS retrieval is published in [6].

2. REQUIREMENTS

Various collaborative filtering mechanisms that allow to retrieve products or services that are of interest to a consumer [9] have been proposed. Those mechanisms are very effective in many domains and seem to be very promising in the context of our work. However, we argue that the multi-faceted nature of SWS imposes special requirements on the underlying feedback mechanism, that are only partially met by existing CF solutions. In the following, we will specify those requirements.

Consumer feedback is *subjective*, since it reflects a service's suitability as perceived through a certain consumer's eyes. Hence, feedback is biased by personal expectations and preferences about the invoked service. Moreover, feedback may refer to different services and to different request contexts. For example, a ticket booking service might have been used to buy group tickets for a school class or to buy a single ticket. However, the suitability of a service might differ depending on the request context and hence the resulting feedback also does. Feedback mechanisms should account for those facts. To enable effective usage, feedback has to be *meaningful*, i.e., the expectations and the context underlying a judgment should be clear. In addition, it should be evident whether and how feedback made under one circumstance can be used to infer about a service's suitability in another situation.

We would also like to emphasize the necessity of feedback to be as *detailed* as possible, i.e. comprising of judgments referring to various aspects of a service interaction. This is for several reasons. Firstly, feedback, judging the quality of a provided service as a whole, is of limited significance, since as an aggregated judgment it provides not more than a rough estimate of a service's performance. Secondly, aggregated feedback tends to be inaccurate. This is due to the fact, that humans are bad at integrating information about different aspects, as they appear in a multi-faceted service interaction, in particular if those aspects are diverse and incomparable [2, 10]. Finally, it has been shown in [4] that using detailed consumer feedback allows to estimate user taste's more accurately and thus can significantly improve prediction accuracy. In the context of detailed, i.e. multi-criteria, consumer feedback, meaningful also means that the relationship between different service aspects that might have been judged is clear and that all relevant aspects characterizing a certain service interaction have been judged. The latter is due to the fact, that inferred judgments based on incomplete information might be incorrect.

Another problem we encounter is feedback *scarcity*. Given certain service requirements, a certain context and a particular service, feedback for exactly this set-up is rare and typically not available at all. Hence, scarce feedback has to be exploited effectively. In particular, service experiences related to different, but similar contexts and those related to other, but similar services have to be leveraged. However, unfolding the full potential of consumer feedback, in particularly when using multi-aspect feedback, requires that users provide useful responses. To ensure this, the feedback elicitation process should be assisted. In particular, it should be taken care that elicited feedback is comprehensive and *appropriate* in the context of a certain service interaction. In addition, a consumer's *willingness* to provide feedback as well as his *expertise* in the service domain should be accounted for. This is important, since asking a consumer for a number of judgments he is not able and/or not willing to provide will result in no or bad quality feedback. Finally, it should also be ensured that all relevant information that are necessary for effectively exploring consumer feedback are recorded. This should happen transparently for the user.

Since the type of service interactions to be judged and the kind of users that provide feedback are diverse and not known in advance, even for a specific area of application, a hard-wired solution with predefined service aspects to judge is inappropriate. In fact, the process of feedback elicitation should be *customizable* and should be *automatically configurable* at runtime.

3. RELATED APPROACHES

Aspects such as feedback scarcity and subjectivity of consumer feedback are typically addressed in existing collaborative filtering solutions [9]. Also, dealing with the context-dependent nature of judgments has been an issue (see e.g. [1]). However, existing solutions only partially address the question of how to effectively use judgments made in one context to infer about a service's suitability in another context. Multi-criteria feedback has been an issue in both academic [4] and commercial recommender systems. Typically, the set of aspects that might be judged by a consumer is either the same for all product types or specific per product category. However, in the first case, this set of aspects is either very generic, i.e. not product-specific, or not appropriate for all products. In the second case, this set has to be specified manually for each new product. Moreover, typically the single aspect ratings are supplementary in the sense that they do not have any influence on a product's overall rating. Alternatively, some reviewing engines such as those provided by Epinions [1]or Powerreviews[2], offer more flexible reviewing facilities based on tagging. Those systems allow consumers to create tags describing the pros and contras of a given product. These tags can then be reused by other users. Tagging provides a very intuitive and flexible mechanism that allows for product-specific judgments. However, the high flexibility of the approach is at the cost of the judgments' meaningfulness. This is due to the fact that tags do not have a clear semantics. In particular, the relationship between different tags is unknown and thus makes them incomparable. Moreover, those systems do not ensure that all relevant aspects of a product or a service interaction are judged. To summarize our findings, more flexible and adaptive mechanisms to elicit and describe multi-criteria feedback are required. In particular, the question of how to describe this type of feedback meaningfully has been hardly considered. To the best of our knowledge, the issue of assisting consumers in providing comprehensive, appropriate and meaningful feedback has not been addressed at all. Also, aspects such as a consumer's ability and willingness to provide judgments for specific aspects have been hardly considered in existing solutions.

4. SEMANTIC WEB SERVICE RETRIEVAL

As a basis for further discussion, we introduce the semantic service description language DSD (DIANE Service Description) [7] and its mechanisms for automatic semantic service matchmaking that underlie our approach. Similarly to other service description approaches, DSD is ontology-based and describes the functionality a service provides as well as the functionality required by a service consumer by means of the precondition(s) and the set of possible effect(s) of a service execution. In the service request depicted in Fig. 1, the desired effect is that a product is owned after service execution. A single effect corresponds to a particular service instance that can be executed. While service offer descriptions describe the individual service instances that are offered by a service provider, e.g. the set of mobile phones offered by a phone seller, service request descriptions declaratively characterize the set of service instances that is acceptable for a consumer. In the service request

[1]http://www.epinions.com
[2]http://www.powerreviews.com

in Fig. 1, acceptable instances are mobile phones that are cheaper than 50$, are either silver or black, are of bar or slider style and are from either Nokia or Sony Ericsson. As

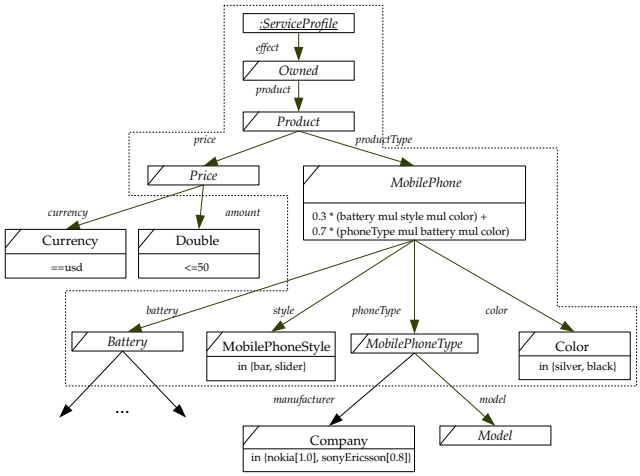

Figure 1: DSD service request

can be seen in the example, DSD utilizes a specific mechanism to declaratively and hierarchically characterize (acceptable) sets of service effects: Service effects are described by means of their attributes, such as price or color. Each attribute may be constrained by direct conditions on its values and by conditions on its subattributes. For instance, the attribute phoneType is constrained by a direct condition on its subattribute manufacturer, which indicates that only mobile phones from Nokia or Sony Ericsson are acceptable. The direct condition $<= 50$ on the price amount in Fig. 1 indicates that only prices lower than 50$ are acceptable. Attribute conditions induce a tree-like and more and more fine-grained characterization of acceptable service effects. A DSD request does not only specify which service effects are acceptable, but also indicates to which degree they are acceptable. In this context, a preference value from $[0, 1]$ is specified for each attribute value. The default is 1.0 (totally acceptable), but alternative values might be specified in the direct conditions of each attribute. For example, the preference value for the attribute manufacturer is 1.0 for Nokia phones and 0.8 for mobile phones from Sony Ericsson.

As demonstrated in [7], DSD service and request descriptions can be efficiently compared. Given a service request, the semantic matchmaker outputs an aggregated overall preference value $\in [0, 1]$ for each available service offer description. This value is called *matching value* and indicates how ell a considered service offer fits to a consumer's requirements encoded in the service request. Based on the matching values, the best fitting service offer is determined and invoked.

5. FEEDBACK ELICITATION

In the following, we will analyze what is required to make detailed consumer feedback meaningful, comprehensive and appropriate to characterize a certain service interaction. We will demonstrate how semantic service descriptions can be used to elicit feedback that fulfills those requirements. A detailed discussion on how to effectively use the elicited consumer feedback to enhance SWS retrieval is out of the scope

of this paper and is published in [6].

What is required to make consumer feedback appropriate, comprehensive and meaningful.

We assume, that a service request at least covers all service aspects that are important to the consumer. Potentially, all service aspects in a request description might be rated by a consumer. In order to be able to exploit these ratings, we need to make sure that they are meaningful (i.e., contain the rating context, e.g., which product a rating refers to) and comprehensive (i.e., contain all relevant aspects, a quality rating without information whether the price was ok is not helpful). In addition, we need to know how different service aspects relate to each other (e.g., how can a rating about quality be gained from ratings on subaspects such as usability and battery capacity?). The challenging question is how to fulfill the identified requirements while still being flexible in the choice of the aspects to rate.

Creating appropriate, comprehensive and meaningful consumer feedback.

We propose the concept of a *feedback structure* to deal with that issue. A feedback structure is a subtree of the request tree, whose leaves correspond to the aspects that may be rated by the user. Consider the example request depicted in Fig. 1. The dotted part of the tree indicates a possible feedback structure for that request, where the aspects price, battery, style, color and phoneType have to be rated by the consumer. Note that this structure contains all information that are necessary to effectively utilize the provided ratings. In particular, it encodes the context of a rating in terms of the path from the request root to the rated aspect, the other aspects that were judged and the hierarchical relationship between the considered aspects.

To assure that the provided feedback is comprehensive, the request subtrees rooted at the feedback structure's leaves should cover all leaves of the tree. This guarantees that all service aspects considered in the request description are either directly or indirectly (by providing an aggregated rating) judged by the service consumer. The feedback structure depicted in Fig. 1 fulfills this requirement and thus is valid. Omitting, e.g., the aspect phoneType would result in an invalid structure. Note, that we are still flexible in the choice of the attributes to be rated, e.g. we could allow the consumer to provide a single rating for productType instead of asking him to judge battery, style, color and phoneType separately. The feedback structure together with the consumer provided ratings are propagated to other consumers and might be used to infer knowledge about a service's suitability for consumers with other service requirements (see [6] for details).

6. RECOMMENDING WHAT TO JUDGE

To ensure feedback quality, the feedback elicitation process should be assisted and should account for a consumer's judgment preferences such as his willingness to provide ratings as well as his expertise in the considered service domain. However, those judgment preferences might differ from request to request, e.g. I might be an expert in judging the quality of Personal Computers, but I do not know that much about servers. As a consequence, I like to/I'm able to judge the quality of a purchased computer, in case of a PC, but I'm not willing to do that when purchasing a new server for our working group. This aspect should be considered during feedback elicitation. To achieve this, we propose the following solution.

Assume, that given a certain service request, an appropriate service was selected and invoked and now its suitability has to be judged by the consumer. In a first step, we utilize the provided service request to determine possible feedback structures as defined in the previous section. Subsequently, the structure that is most suitable for the user, i.e. in the context of the given request, fits best to the consumer's personal abilities and judgment preferences, is selected and presented to the user. The required knowledge about the user's judgment requirements is learned from the his behavior in previous judgment sessions. The presented feedback structure represents a careful compromise between the consumer's competing judgment requirements and might be adjusted to his actual judgment needs. This can be done by expanding and/or hiding subtrees of the presented structure. For example, in the structure depicted in Fig. 1, we might expand the leaf phoneType to judge its subaspects manufacturer and model. Finally, the user judges all leaf attributes of the structure, e.g. by providing a rating. Once, the consumer submits his judgments, the system takes care of storing all relevant feedback information and session data for future recommendations. In particular, it is recorded which and how many service aspects were judged by the consumer and which service request lead to the judgment. The acquired information are used later on to identify suitable feedback structures in future judgment sessions.

6.1 Feedback structure suitability

Given a consumer's service request, typically many different feedback structures are possible. However, how to measure the suitability of each feedback structure to identify one that fits best to the user's personal abilities and willingness to provide judgments? We have to consider two aspects here. Firstly, comprise the feedback structure leaves of those attributes that the consumer's is able to judge and secondly, is the consumer willing to judge all those aspects?

As a measure of a consumer's willingness and ability to judge a certain service aspect, we us the frequency with which the user judged this aspect in the past. We also consider the request context in which an aspect was judged. More specifically, we consider how similar the request that lead to the past judgment is to our request. Let r be the service request that was posed by the consumer. Then the consumer's willingness and ability to judge service aspect a is determined by $w_a(r) = \sum_{r' \in R_a} sim(r', r)$, where R_a is the set of past service requests that lead to a judgment of a. The value $sim(r', r)$ indicates how similar the service requirements encoded in the past request r' are to those in current request r. A detailed discussion on how compute the semantic similarity of two requests is provided in Sect. 6.3.

The suitability $s_{attributes}(fs, r)$ of a given feedback structure *fs* is determined by the consumer's willingness and ability to judge its leaf aspects A_{fs}. We propose to compute it as the sum of its leaf attributes' w_i-values.

$$s_{attributes}(fs, r) = \frac{\sum_{i \in A_{fs}} w_i(r)}{\sum_{j \in A_r} w_j(r)} \quad (1)$$

The term is normalized by dividing it by the sum of the w_j-values of all attributes $j \in A_r$ that are contained in the

given request r. Hence, $s_{attributes}(fs, r) \in [0, 1]$.

To measure a consumer's willingness to judge $k = |A_{fs}|$ leaf aspects A_{fs}, we compare how similar the past requests that also led to a judgment of k aspects are to the service request r posed by the consumer. More specifically, the suitability $s_{number}(fs, r)$ of the feedback structure fs with respect to the number of service aspects that have to be judged is determined by

$$s_{number}(fs, r) = \overline{sim}(R_k, r), \tag{2}$$

where $\overline{sim}(S_k, r)$ is the mean request similarity of all past service requests that lead to a judgment of k aspects. In cases, where no previous requests lead to a number of k service aspects to be judged, $s_{number}(fs, r)$ is determined as the mean of $\overline{sim}(R_{k'}, r)$ and $\overline{sim}(R_{k''}, r)$, where k' is the largest $k' < k$ for which a past request with k' judgments exists and k'' is the smallest $k'' > k$ for which a past request with k'' judgments exists. In case, k'/k'' did not exist, $\overline{sim}(R'_k, r)/\overline{sim}(R''_k, r)$ was assumed to be 1.0/0.0, i.e. by default feedback structures with a low number of service aspects to be judged are preferred. Assuming that $\overline{sim}(x, y)$ is a value from $[0, 1]$, $s_{number}(fs, r)$ is also from $[0, 1]$. The overall suitability $s(fs, r) \in [0, 1]$ of a feedback structure fs in the context of the posed request r is

$$s(fs, r) = \alpha \cdot s_{attributes}(fs, r) + \beta \cdot s_{number}(fs, r). \tag{3}$$

The parameters α and β with $\alpha, \beta \in [0, 1]$ and $\alpha = 1 - \beta$ determine the influence of the terms $s_{attributes}(fs, r)$ and $s_{number}(fs, r)$, respectively. The values α and β might vary from user to user. In Sect. 6.4, we will demonstrate how those values can be learned from a consumer's past judgment behavior.

6.2 Determining possible feedback structures

For a given request, the number of possible feedback structures might be high, whereas the number of those that have the potential to be optimal (with respect to their suitability $s(fs, r)$ for the user) is low. Hence, we require a way to determine potentially optimal feedback structures effectively, i.e. without having to construct all possible structures. In the following, we propose an algorithm that performs this task. It constructs potentially optimal feedback structures recursively and drops non-optimal partial structures as soon as possible. Fig. 2 shows how the algorithm

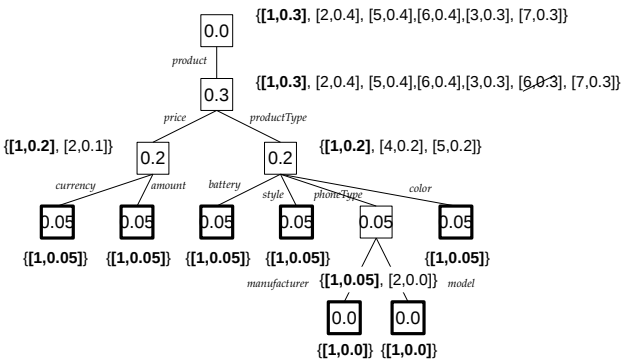

Figure 2: Determining possible feedback structures

works, exemplary for the service request depicted in Fig. 1. Each request node is associated with a list of entries, each corresponding to one of the feedback structures that are possible for the subtree rooted at that node. Let fs be one of those structures and let $[a, b]$ be its corresponding entry. Then a is the number of aspects that have to be judged in fs and b is $s_{attributes}(fs, r)$, where r is the request subtree rooted at the considered node. The algorithm works as follows. First, it initializes each request node's list with an entry $[1, s_{attributes}(fs, r)]$, where fs is the feedback structure comprising only of the node itself and r is the request subtree rooted at the considered node. For an example, consider Fig. 2. The initial entry in each list is highlighted. The number within each node indicates the value $s_{attributes}(fs, r)$, which, for the sake of this example, is arbitrarily chosen. Starting from the request leaves (highlighted request nodes), the algorithm recursively computes lists for all parent nodes. Computing a node's list is done in three steps. First, the cross product C of the child nodes' entry sets is computed. For example to determine possible feedback structures for the product-node (Fig. 2), we have to determine $C = [1, 0.2], [2, 0.1] \times [1, 0.2], [4, 0.2], [5, 0.2]$, i.e. the cross product of the price and productType node's entry lists. Each element c of C gives rise to an entry $[a, b]$ in the product-node list, i.e. to a possible feedback structure fs of this node's subtree. Since a is the number of attributes to judge in fs, it is computed as the sum of the a values in c. The suitability b of fs with respect to the selection of attributes that have to judged is computed as the sum of its leaf attributes' b values (Formula 1), i.e. the sum of the b values in c. In a final step, we prune the computed list. This is done by keeping only a single entry $[a, b]$ for each different value of a per node, where $b = \max\{x | [a, x] is in the list\}$. Note, that in doing so, we keep only those feedback structures that have the potential to be optimal and hence reduce the length of the node list to at most l, where l is the number of leaves of the subtree rooted at the considered node. Finally, we end up with a list for the request root comprising of entries for all possible feedback structures for the request, that have the potential to be optimal. Those structures are compared with respect to their suitability (Formula 3). The most suitable is selected and presented to the user.

6.3 Request similarity

As mentioned earlier, a consumer's judgment preferences depend on the request context, i.e. the kind of service interaction, that has to be judged. To allow for a comparison of the request contexts, in which judgments have been made in the past, with the current request, we require a measure for the semantic similarity of two requests, i.e. the similarity of the service requirements they encode. In this section, we will propose such a measure. It recursively computes the similarity $sim(r, r')$ of two request trees r and r' by computing the similarity of their root nodes' ontological type $(sim_{type}(root(r), root(r')))$ and direct conditions $(sim_{dc}(root(r), root(r')))$ and the aggregated similarity of their root nodes' child trees $(sim_{attr}(root(r), root(r')))$. More specifically, we define $sim(r, r')$ to be the mean of these three values. In the remainder of the section, we will explain the rationale between those three similarity values and particularize on how to determine them. Possible similarity values $sim(r, r')$ are from the interval $[0, 1]$, where a similarity value of 0.0 means "not similar at all" and a value of 1.0 means that the service requirements encoded by two requests are

identical.

Determining the type similarity.

The type similarity $sim_{type}(n, n') \in [0, 1]$ of two nodes n and n' indicates how similar those nodes are with respect to their ontological type. It is defined similar to Jaccard's index [5], that is often used to compare the sample sets,

$$sim_{type}(n, n') = \frac{|A_n \cap A_{n'}|}{|A_n \cup A_{n'}|} \quad (4)$$

where A_n is the set of attributes defined for the type of n and $A_{n'}$ is the set of attributes defined for the type of n'. The type similarity $sim_{type}(n, n')$ for the root nodes of the requests depicted in Fig. 3 is $\frac{|\{battery, phoneType, color\}|}{|\{battery, phoneType, color, style\}|} = 0.75$.

Determining the similarity of the direct conditions.

The similarity $sim_{dc}(n, n') \in [0, 1]$ of two nodes n and n' indicates how similar those nodes are with respect to their direct conditions. As mentioned in Sect. 4, direct conditions restrict acceptable values of a service attribute. For each kind of direct condition that might be specified for a certain attribute, we define a separate similarity measure. For example, for direct conditions of type $IN\{\ldots\}$, the similarity is determined as the quotient of the number of common values divided by the number of values that are allowed for n or n'. For direct conditions of type $<= x$ and $>= x$, the similarity is calculated as $\min\{x, y\}/\max\{x, y\}$, where x is the upper/lower bound for the values of n and y for those of n'. Accordingly, if only one of the nodes specified a certain type of direct condition, the similarity is defined to be 0.0 and if both nodes do not specify any direct conditions, the similarity is defined to be 1.0.

As an example, consider again the requests depicted in Fig. 3. The Color-nodes both specify a direct condition of type $IN\{\ldots\}$. The similarity $sim_{dc}(n_{color}, n'_{color})$ with respect to this direct condition is $1/2 = 0.5$. The Battery-nodes both do not specify any direct conditions, hence $sim_{dc}(n_{battery}, n'_{battery}) = 1.0$.

Determining and aggregating the similarity of the root nodes' child trees.

The similarity value $sim_{attr}(n, n') \in [0, 1]$ indicates how similar two nodes n and n' are with respect to their child trees. Let A be the set of attributes defined either for the type of n, the type of n' or for both types and let $\{sim(r_a, r'_a) | a \in A\}$ be the similarity values for corresponding attribute subtrees r_a and r'_a of n and n'. Again, inspired by Jaccard's index, the aggregated similarity of two nodes' child trees is defined as the sum of the similarity values $\{sim(r_a, r'_a) | a \in A\}$ divided by the sum of the maximal similarity values that can be achieved for each attribute, i.e. $|A|$.

$$sim_{attr}(n, n') = \frac{\sum_{a \in A} sim(r_a, r'_a)}{|A|} \quad (5)$$

Since attributes in A are not necessarily defined for both, the type of n and n', we set $sim(r_a, r'_a) = 0.0$, if the attribute a is not defined for one type. Attributes in A might also not be specified in one or both of the nodes. If an attribute a is not specified in both nodes, we set $sim(r_a, r'_a) = 1.0$, else, if a is specified in just one of the nodes, $sim(r_a, r'_a)$ is defined to be $sim(r_a, t')$ resp. $sim(t, r'_a)$, where t is a tree comprising of a single node, having the most generic type defined for a in the ontology. We illustrate the procedure for the root nodes of the two request fragments depicted in Fig. 3. The type of r's root node is MobilePhone and that of r''s root node is Phone. Assume, that the ontology defines the attributes battery, phoneType and color for the type Phone and an additional attribute style for the type MobilePhone, which is a subtype of Phone. The similarity $sim_{attr}(n, n')$ of the requests' root nodes n and n' is determined by the similarity of their corresponding child trees for the attributes $A = \{$battery, phoneType, color, style$\}$. The attributes battery and color are specified in both requests, hence the similarity values $sim(r_{battery}, r'_{battery})$ and $sim(r_{color}, r'_{color})$ can be computed by determining the request similarity for the request subtrees rooted at the Battery-nodes and the subtrees rooted at the Color-nodes. The attribute style is only defined for the type MobilePhone, hence $sim(r_{style}, r'_{style}) = 0.0$. The attribute phoneType is defined for both types, MobilePhone and Phone, but only specified in r. Hence, $r'_{phoneType}$ has to be replaced by a node t' having the most generic type defined for the attribute phoneType. Let PhoneType be this type. This means that the type of the node that describes the attribute phoneType has to be PhoneType or one of its subtypes. Presume that MobilePhoneType is a subtype of PhoneType. The similarity $sim(r_{phoneType}, r'_{phoneType})$ is determined by computing $sim(r_{phoneType}, t')$, where $r_{phoneType}$ is the subtree of r rooted at the MobilePhoneType-node.

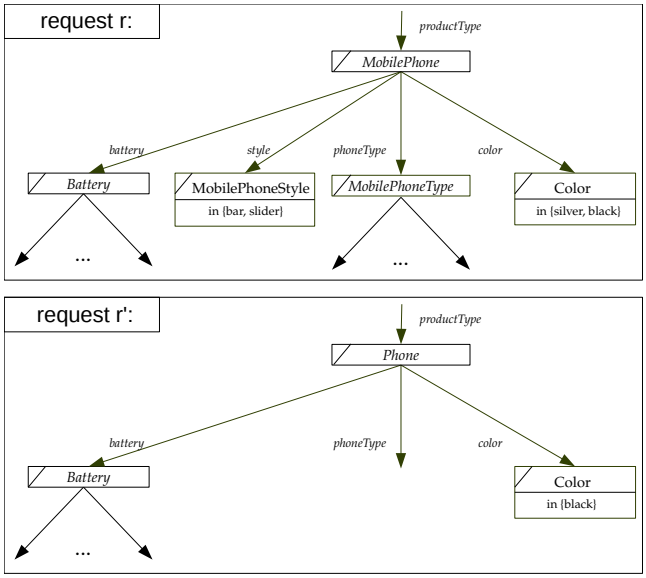

Figure 3: Determining the similarity of two requests r and r'

6.4 Dynamically adjusting α and β

As discussed earlier, the parameters α and β that weight the influence of the terms $s_{attributes}(fs, r)$ and $s_{number}(fs, r)$, might vary from user to user. In this section, we will demonstrate how those values can be learned from a consumer's past judgment behavior. Initially, i.e. without having information about a user's previous judgment behavior, we do not know anything about those parameters' values, so α could be any value from the interval $[0, 1]$ and $\beta = 1 -$

α. Hence, for the purpose of computing the suitability $s(fs, r)$ of possible feedback structures, we set α to the midpoint of this interval, i.e. $\alpha = 0.5 = \beta$. Once having determined the most suitable feedback structure *fs*, we present it to the consumer, who has the opportunity to change it by expanding/collapsing nodes. Finally, the consumer provides judgments for the resulting structure's leaf nodes. Obviously, the resulting feedback structure *fs'* was more suitable to the user than the structure *fs* that was recommended. Hence, we conclude that $s(fs', r)$ should be larger than $s(fs, r)$. Using Formula 3, we get that $s(fs, r) - s_{number}(fs', r)/s_{attributes}(fs', r) - s_{number}(fs', r) < \alpha$ for $s_{attributes}(fs', r) > s_{number}(fs', r)$ and $> \alpha$ for $s_{attributes}(fs', r) < s_{number}(fs', r)$. Using those information, we can adjust, i.e. shrink the range of α correspondingly. For example, if we get $\alpha < 0.8$, we adjust the interval to $[0, 0.8)$. In case, the consumer's judgment behavior is inconsistent, e.g. having $\alpha \in (0.5, 0.7)$, we get $\alpha < 0.8$, we simply ignore those information. To ensure, that the most recent information have the most influence, we process session data in the order of increasing age.

7. EVALUATION

In the evaluation of our approach, we wanted to find out how fast the recommendation algorithm proposed in Sect. 6 adjusts to different judgment preferences.

Test setting.

For that purpose, we created a set of DSD service requests covering typical requirements of consumers looking for computer items from different categories, such as desktop PCs, PDAs, servers, notebooks or organizers. For our tests, we created 48 service requests, 6 per category. Requests within each category varied in the selection of attributes that were specified and in the range of attribute values that were acceptable for the user. All request types shared common attribute types, e.g. for all kinds of requests an attribute color and an attribute price could be specified.

Using this requests we performed several tests with a single test user. The basic procedure for each test was as follows. Starting with no information about previous judgment behavior, several judgment sessions were performed. During each session, one of the 48 requests was selected. After that, the system proposed a feedback structure using the algorithm proposed in Sect. 6 with knowledge about the user's judgment behavior in the previous judgment sessions. After being provided with the recommended feedback structure, the user had the opportunity to change this structure. For that purpose, the consumer was allowed to expand/collapse feedback structure nodes. By clicking on a particular node, all its direct children were expanded/collapsed. The quality of the proposed feedback structure was measured as the edit distance between the proposed feedback structure and the actual feedback structure that was used. More formally, we counted the number of expand/collapse operations the user had to perform to get the structure whose leaves he finally judged. The rationale behind this measure is, that the edit distance is a direct measure of the users effort to get to the desired structure and thus, in our opinion, is a good measure for the quality of the recommended structure. For each of the test, we looked at whether and how fast the edit distance decreased with the number of judgment sessions.

Test runs and results.

We performed test runs with different judgment preferences and different sets of requests that were posed during a sequence of sessions. In a first series of tests, the requests within each sequence of sessions were different, but chosen from a single (computer) category, e.g. just notebook requests. This test setting served as a baseline and was chosen to evaluate the performance of our approach in the absence of any context effects. We performed three kinds of tests differing in the judgment preferences of the judging user. In test $A1$, the consumer always judged a certain number of aspects. However, the types of aspects that were judged differed. In test $A2$, the user judged a different number of attributes during each session, but required that the set of attributes to judge contained a certain set of attributes. For example, a user might require to always judge the price of a product, but is also willing to rate other service aspects. Finally, we performed a test $A3$, were the consumer had specific requirements on both, the number and kind of aspects to judge. The tests $A1$-$A3$ were performed with request sets from different categories. The plot depicted in Fig.4 (A2) is representative for all test runs and all types of tests in this series. It shows the results for test $A2$ performed with requests from the category digital watches. As can be seen, the adaptation of the recommendation algorithm to the consumers judgment preferences is very fast. The initial edit distance decreases to 0 after just one session. This is due to the fact, that request similarity does not play a role in those tests and hence the values of α and β can be arbitrarily chosen. The depicted behavior was observed for all three kinds of tests ($A1$-$A3$).

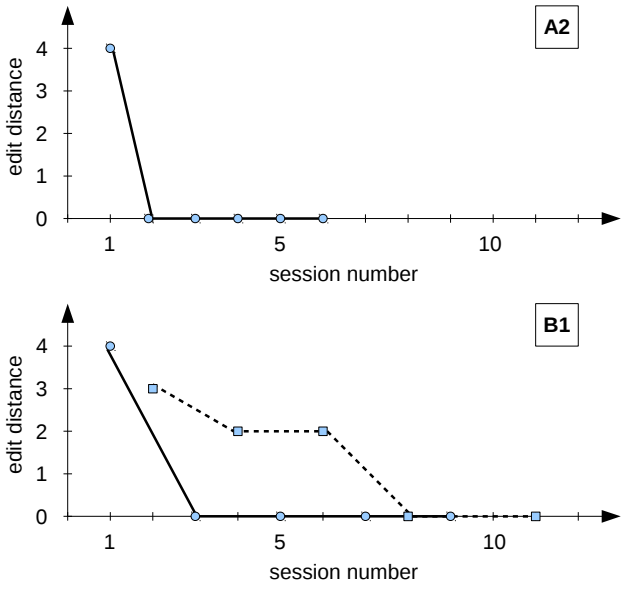

Figure 4: Results of the tests $A2$ and $B1$

In a second series of tests, we evaluated how fast the proposed recommendation algorithm adjusts to a consumers judgment preferences, if those depend on the request context. For that purpose, we performed judgment sessions, were the user posed requests from different categories and exhibited a different judgment behavior for each category. We run three types of tests. In test $B1$, similarly to test

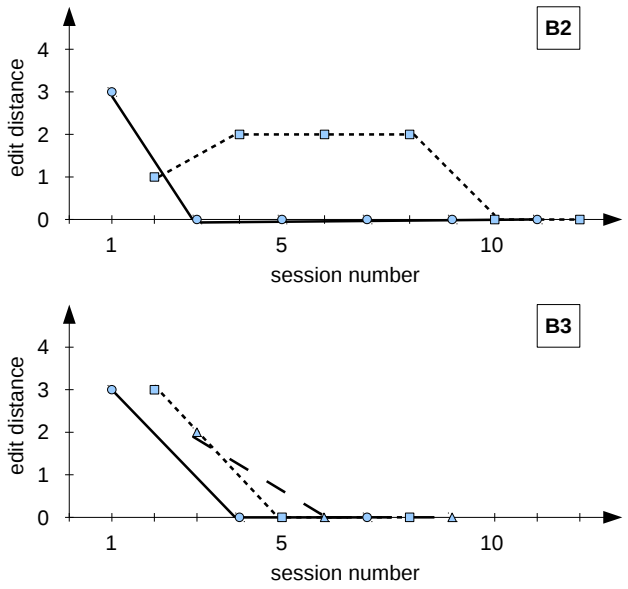

Figure 5: Results of the tests $B2$ and $B3$

$A1$, the user always judged a particular number of aspects. However, this number differed for each request category. For example, a user might always judge 3 aspects when asking for desktop PCs, but is willing to judge 5 service aspects, when asking for notebooks. Analogously to test $A2$, the user in test $B2$ required the set of aspects to be judged to contain a particular set of aspects. However, this set varied for different request categories. Finally, we performed a test $B3$, were the consumer had specific requirements on both, the number and kind of aspects to judge. Those requirements were different for each request category. Fig. 4 (B1) exemplary shows the results for tests of type $B1$. In the depicted test, we alternated sessions based on a request for a desktop PC (continuous line), where the user judged 11 service aspects, with those based on a request for a PDA (dotted line), where the consumer judged only one aspect. As can be seen, the adjustment to the consumer's judgment preferences for PDAs takes 3 sessions. This is due to the fact, that at the beginning both terms $s_{attributes}$ and s_{number} are equally weighted. Since for desktop PCs many aspects are judged and since most of those aspects are also shared by PDA requests, term $s_{attributes}$ dominates the suitability value and thus favors improper feedback structures. This changes when α and β adjust over time. Fig. 5 (B2) exemplary shows the results for tests of type $B2$. Again, we alternated desktop PC requests with those for a PDA. While when judging desktop PCs, we had a set of two aspects that had to be judged in any case, it was only one specific aspect when judging PDAs. Again, it required 4 sessions to adjust α and β appropriately. Finally, Fig. 5 (B3) exemplary shows the results for tests of type $B3$. In this test, we alternated three types of requests (desktop PC, PDA and digital watch requests). As can be seen, the algorithm propose appropriate feedback structures after just 1 session of each type. This is due to the fact, that for the three request types, the consumer's judgment behavior differed much in terms of the number and types of aspects to be judged. Hence, though α and β are not yet adjusted, the correct feedback structure can be identified.

8. CONCLUSION

In this paper, we demonstrated how detailed consumer feedback, that is meaningful and appropriate in the context of a service interaction, can be elicited and how users can be supported in that process. Our main contribution is an algorithm that suggests service aspects that might be judged by a consumer. Our evaluation results show, that the proposed procedure effectively adjusts to a user's ability and willingness to provide judgments.

9. REFERENCES

[1] G. Adomavicius. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103, 2005.

[2] R. M. Dawes. The robust beauty of improper linear models in decision making. *American Psychologist*, 34(7):571–582, 1979.

[3] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer, 2007.

[4] Y. K. Gediminas Adomavicius. New recommendation techniques for multi-criteria rating systems. *IEEE Intelligent Systems*, 22(3), 2007.

[5] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[6] F. Klan and B. König-Ries. Enabling trust-aware semantic web service selection - a flexible and personalized approach. Jenaer Schriften zur Mathematik und Informatik, Math/Inf/02/10, Friedrich-Schiller-University Jena, August 2010.

[7] U. Küster, B. König-Ries, M. Klein, and M. Stern. Diane - a matchmaking-centered framework for automated service discovery, composition, binding and invocation. In *Proceedings of the 16th International World Wide Web Conference (WWW2007)*, Banff, Alberta, Canada, May 2007.

[8] U. S. Manikrao and T. V. Prabhakar. Dynamic selection of web services with recommendation system. In *Intl. Conf. on Next Generation Web Services Practices*, pages 117–121, Washington, DC, 2005. IEEE Computer Society.

[9] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 291–324. Springer, Berlin, Heidelberg, 2007.

[10] P. Slovic. *Limitations of the Mind of Man: Implications for decision making in the nuclear age*. Los Alamos Scientific Laboratory, 1972.

[11] H. C. Wang, C. S. Lee, and T. H. Ho. Combining subjective and objective qos factors for personalized web service selection. *Expert Syst. Appl.*, 32(2):571–584, 2007.