Agent-Based Service Selection ⋆

Raghuram M. Sreenath [a] Munindar P. Singh [b,*]

[a] *IBM, 3039 Cornwallis Road, Research Triangle Park, NC 27709, USA*

[b] *Department of Computer Science, North Carolina State University, Raleigh NC 27695, USA*

Abstract

The current infrastructure for Web services supports service discovery based on a common repository. However, the key challenge is not discovery but *selection:* ultimately, the service user must select one good provider. Whereas service descriptions are given from the perspective of providers, service selection must take the perspective of users. In this way, service selection involves pragmatics, which builds on but is deeper than semantics.

Current approaches provide no support for this challenge. Importantly, service selection differs significantly from product selection, which is the problem addressed by traditional product recommender approaches. The assumptions underlying product recommender approaches do not hold for services. For example, a vendor site knows of all product purchases made at it, whereas a service registry does not know of the service episodes that may involve services discovered from it. Also, traditional approaches assume that users are willing to reveal their evaluations to each vendor site.

This paper formulates the problem of service selection. It reformulates two traditional recommender approaches for service selection and proposes a new agent-based approach in which agents cooperate to evaluate service providers. In this approach, the agents rate each other, and autonomously decide how much to weigh each other's recommendations. The underlying algorithm with which the agents reason is developed in the context of a concept lattice, which enables finding relevant agents.

Since large service selection datasets do not yet exist, for the purposes of evaluation, we reformulate the well-known product evaluations dataset MovieLens as a services dataset. We use it to compare the various approaches. Despite limiting the flow of information, the proposed approach compares well with the existing approaches in terms of some accuracy metrics defined within.

Key words: service selection, agents, recommendation and rating systems

1 Introduction

In traditional closed environments, programmers manually configure their software systems so that the desired components can be linked in and used. This task proves quite onerous in large-scale distributed systems, especially when the components are heterogeneous (of differing and largely unknown design and construction) and autonomous (representing different interests and not necessarily acting in a trustworthy manner). In general, heterogeneity and autonomy are the defining characteristics of modern open information environments.

Web Services The emerging Web services approach partly addresses the above problem. Software components are modeled as services. Service providers publish descriptions (such as using the Web Services Description Language (WSDL) [WSDL, 2002]) of the services that they support to one or more registries (following the Universal Description Discovery and Integration (UDDI) registry standard [UDDI, 2002]). A prospective service consumer can query a registry to obtain information for service implementations that meet a specified description. This is clearly a great help in the case of a large-scale system, because it disseminates information about the available components to those who would wish to use such components in their software systems. In fact, the programmer is no longer needed in the critical path, because services can be found and bound dynamically and automatically by the software system.

Let's examine how the above Web services approach addresses the problem of configuring distributed systems in large-scale open information environments. First, configuration involves a decision to select a particular service implementation. The traditional approach offers no support for this decision. However, we can consider how it would deal with the heterogeneity and autonomy of the services being bound. The heterogeneity aspect of configuration is handled well, because we do not need to know about the internal construction of the Web services. And, additional semantics can potentially be accommodated (see below). By contrast, the autonomy aspect of configuration is not handled well, because there is no support for expressing whether the autonomous behavior of a service is desirable and no means to influence the configuration process.

⋆ This research was supported by the National Science Foundation under grant ITR-0081742.

* Corresponding author
Email addresses: sreenath@us.ibm.com (Raghuram M. Sreenath), singh@ncsu.edu (Munindar P. Singh).

Semantic Web Services Richer representations of Web services are being developed. A major class of efforts is motivated from the Semantic Web activities [Berners-Lee et al., 2001; Hendler and McGuinness, 2001]. Some approaches add further structure to service descriptions through the use of ontologies, e.g., [Trastour et al., 2001]. DAML-S [DAML-S, 2002]) is based on DAML, the DARPA Agent Markup Language. DAML-S provides an upper ontology and core set of markup language constructs for describing the properties and capabilities of Web services. It enables searching for services according to their functionality. DAML-S' main contribution is in specifying the structure of the processes through which services are implemented.

Architecturally, the semantic approaches fit well into the current Web services approach, because the current strategy of publishing service description to a repository and matching descriptions are naturally extended to accommodate semantic descriptions. Besides descriptions, recent research in this area has considered either matchmaking or sophisticated reasoning, e.g., via planning, to compose services [McIlraith et al., 2001]. There is clearly much value in semantic descriptions of Web services, which can enable better matchmaking than syntactic descriptions alone.

Selection The current Web services architecture as well as the current semantic Web enhancements address the problem of service discovery but not of service *selection*. Broadly, discovery deals with finding service implementations that meet a specified description; in essence the discovered services are deemed equivalent in meeting the specified description. In the same vein, selection deals with choosing a service implementation from among those that are discovered for the given description. Discovery is a prerequisite for selection, but it is selection which is the main problem. In fact, being too complete at discovery can only make selection harder. The better a repository becomes in terms of the number of service descriptions published to it the worse it is for selection, because it would produce longer and longer lists of potential services.

We make a distinction between service description and service selection. In order to use Web services successfully, both aspects must be addressed. Service description is handled by DAML-S and other competing standards such as UDDI and WSDL. Service selection is what we address in this paper. Thus, the main aspects of DAML-S (e.g., describing the process models for composed services) are orthogonal to our present topic. In other words, when several services match the given criteria of behavior or functionality, we still need to select the best one. The DAML-S authors acknowledge the importance of automatic service selection. For this purpose, DAML-S includes various *functional attributes*, which include some quality of service parameters. However, the meaning and usage of the functional attributes has not been elaborated

yet. It appears from the informal descriptions that values for these attributes are meant to be supplied by the service providers themselves or by some industry standardization entities. In general, we would expect such evaluations to be best supplied by service consumers rather than by providers; if an industry standard quality measure is available, it too would be based on the consumers. Maximilien and Singh develop a richer conceptual model for quality attributes [2002]. The approach proposed here is able to work with any set of service quality attributes, including when individual attributes of services are not explicit.

For our present purposes, it is clear that regardless of the conceptual model of attributes, the selection of services would usually be based on criteria that are not included in the descriptions posted by the providers. These other, more context-sensitive, criteria are addressed through the concept lattice used in the proposed approach. These criteria relate better to a form of meaning that is pragmatic rather than semantic. Pragmatics can be thought of as a specialization of semantics [Singh, 2002]. That is, selection is a key aspect of semantic Web services that existing work has not addressed adequately.

Contributions This paper makes the following main contributions for service selection. One, it shows how to formulate the problem of service selection building on top of the current Web services architecture. Two, it reformulates existing techniques so they can support service selection, and proposes a new service selection approach based on interactions among agents mediated through an application of concept lattices. Three, it proposes a methodology for evaluating service selection techniques. Four, it shows how the methods considered compare along different metrics.

Organization The rest of this paper is organized as follows. Section 2 introduces service selection in contrast with product selection, discusses current approaches, and gives technical motivation for our proposed approach. Section 3 develops an approach that augments the current Web services architecture with a community-based algorithm to select services. Section 4 describes our experiments, including how we reformulate two existing approaches for services and how we map an existing product dataset to evaluate service selections. Section 5 summarizes our key contributions and conclusions and discusses some directions for future research.

2 Service Selection

In the following, we consider a set of participants who can be *service consumers* (when they request a service) and *service providers* (when they offer to provide a service implementation). For simplicity, we assume that each provider offers no more than one implementation for a given service. Now we consider how a consumer selects one provider.

Service discovery involves mapping a service description to a set of services. In its basic form, service selection involves mapping a set of services to a service—this can be thought of as the best service; in a more general form, service selections maps a set of services to a ranking of the services in that set. At this abstract level, this problem is the same as product selection, which involves mapping a set of products (usually implicit in a vendor's catalog) to a product or a ranking of products. Because of this similarity between all selection tasks, it is appropriate to compare selection techniques on an even footing, whether they be intended for products, services, or anything else. Section 2.1 discusses traditional approaches for selection. However, service selection settings have a number of properties that distinguish them from traditional product settings. Consequently, we need to deviate from traditional product selection techniques as well. Section 2.2 presents further motivation for service selection.

2.1 Traditional Approaches for Selection

It is worthwhile to first consider traditional approaches for product or web-site selection. The traditional approaches of interest include product recommendation, reputation management, peer-to-peer, and referral systems.

2.1.1 Content-Based Filtering

Content-Based Filtering is a static approach for selecting among web-sites (or other kinds of information, such as news items) [Dumais et al., 1988]. It involves filtering web-sites or documents in terms of the keywords that occur in them. This approach could be applied to services by indexing the text descriptions of services based on the words that occur in them, possibly allowing boolean search queries on them. However, this would be a step backward from Web services, which involve formal descriptions and support discovery of services based on those descriptions. Moreover, it would still not address the problem of selection.

2.1.2 Social Information Filtering

Another major family of approaches is *Social Information Filtering*, which refers to the generic technique of making recommendations based on relationships between users and on their subjective judgments. Of these, the most well-known is *Collaborative Filtering (CF)* [Breese et al., 1998; Sarwar et al., 2000]. CF is widely used at established e-commerce sites such as amazon.com. In CF, users' votes for different products are stored centrally—the votes often are simply captured as the purchases made by a given user. A user is given recommendations based on the votes by other users who are similar to the given user. In simple terms, if Alice and Bob both bought books A, B, C, and D, they may be deemed similar. Now if Alice also bought book E, a CF system may recommend that Bob buy book E.

CF works for individual product sites because they know which users purchased what products. However, a service registry that finds matching services for a particular request would not know whether a requester went on to use one of those services and, if so, which one. The distribution of Web services makes CF inapplicable for service selection.

Nakamura and Abe present *generalized learn relationship (GLearn)* as an approach that enables users to combine evidence from other users [1998]. GLearn is a generalized version of the *weighted majority algorithm (WMA)*. WMA is a popular online learning approach, which supports combining multiple predictions [Littlestone and Warmuth, 1994]. Its main idea is to attach weights to several *expert predictors*, each of which makes binary (yes/no) predictions given an instance of the problem. WMA calculates a weighted majority of these predictions and comes up with an overall binary prediction. If the overall prediction is wrong, WMA reduces the weights of all the predictors that voted incorrectly. WMA can be applied by each user to assign weights to other users. Nakamura and Abe generalize WMA to accommodate nonbinary evaluations—we adapt their approach for service selection and compare it to our approach.

2.1.3 Reputation Systems

Commerce sites such as eBay support a mechanism whereby parties who transact can give numeric scores to each other. The scores are aggregated to yield a *reputation*. Reputation systems, e.g., Sporas [Zacharia et al., 1999], come up with a single score for a service provider. In general, service providers with higher reputations would be preferred more.

However, a single aggregate score may not be appropriate for all users. This is because users would have different needs, which may lead them to emphasize different features of the services. What would matter most for a user are the

scores given by users whom the given user has reason to value. Sites such as eBay are closed. Scores given therein apply to eBay transactions only. Other sites, such as *Epinions* [2002], enable users to rate any product or service, but there are no constraints on the scores—the raters may have never used the given product or service. Users can decide which of the other users to trust or block. However, there is no support for fine-tuning these scores.

More importantly, reputation systems provide a coarse-grained way to disseminate evaluations. Scores given by a user are potentially shown to all, whereas the scores received by a user by default originate from whoever cared to publish an evaluation. This contrasts with real life. For example, we would be reluctant to reveal important evaluations to anyone but our friends or to entertain scores given by anyone but our friends.

2.1.4 Peer-to-Peer Systems

The problem of a centralized database is overcome by peer-to-peer (P2P) systems, which are distributed systems with no central authority or management. Searching for a service provider reduces to querying *peers* for the kind of service required. For example, Gnutella one of the most successful P2P systems [Kan, 2001], supports sharing files over the Internet. The system assigns a set of neighbors to each user. When the user requests a file, a query is sent out to all of his neighbors, who in turn may forward the query to their neighbors, and so on until the requested file is found.

Search in P2P systems is geared toward discovery rather than selection. In principle, search can yield multiple services and there is no basis for selecting any of them. This might be acceptable for some users when they wish to download a music file and do not care about the source, but it would matter in general. A major drawback of this approach is that trust is not associated with the suggestions given by the peers. For example, no guarantees are made about the quality of the files that are found via Gnutella.

2.1.5 Referral Systems

A referral system is a kind of a P2P system where the peers not only provide services directly, but may also refer requesters to other peers [Singh et al., 2001]. Each peer is represented by a software agent. These agents maintain a changing list of neighbors, which are their trusted peers. Whenever a service is needed, the agent contacts its neighbors. A neighbor may offer to provide the service or may give referrals to other peers. The referrals act as endorsements. The originating agent uses its rating of a peer to decide whether to select it as a service provider or to follow its referrals. Ultimately, if a suitable service can be found, the agent has a basis for deciding whether to select it. Importantly,

the agent would evaluate the service it ultimately receives. Based on this evaluation, it would change its evaluation of the selected service provider and the peers who gave referrals to that provider.

Referral systems can be potentially quite powerful. However, using them would cause us to deviate from the standard Web services architecture. To maintain practicality, we would like to introduce agents for service selection in a more limited manner that is compatible with the existing architecture.

2.2 Motivation for Collaborative Evaluation

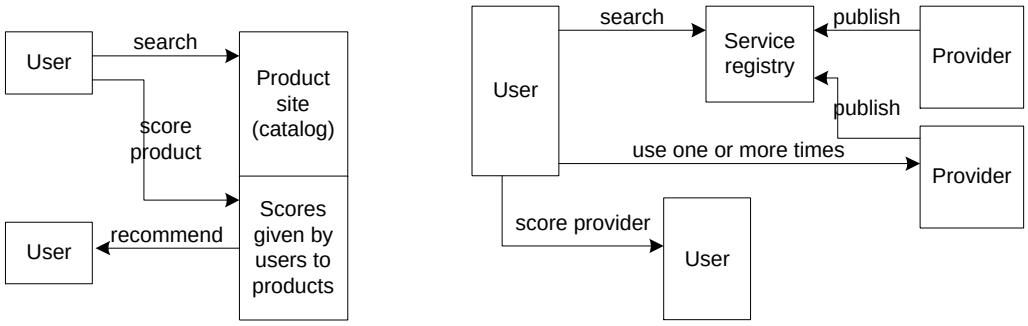

Fig. 1. Product selection (l) and service selection (r)

Figure 1 shows schematics of typical product and service selection. For product selection, a "use product" link is not shown to reduce clutter and because a product is not an entity in the architecture. The scores assigned to products by users may be implicit (often simply capturing whether the user purchased the given product) and are typically stored by the product site; these scores could be stored by independent reputation agencies if they are explicit.

Some key similarities between service and product selection were identified at the beginning of this section. However, service selection is inherently different from product selection for the following reasons. One, services are distributed. In particular, a broker or registry does not itself provide any of the services it lists. That is, the registry helps a service user find a provider, but would not have any control on the actual service invocation and delivery, whereas a product site would know that a particular product was shipped. Two, once a service provider has been discovered and selected, a service user could invoke the provider multiple times. The registry would not even be aware of the repeat users of a provider. Three, because service episodes are richer and more highly context sensitive than product usage, central approaches curtail the autonomy of users to combine scores as they see fit. Even products are distributed, they would not match with services; rather the analogy would be between services and product vendors, which can be thought of as services providing products.

Because of the above properties, it is potentially important for someone receiving a recommendation for a service to identify the source of the information. However, this conflicts with privacy: honest scores would generally not be revealed publicly.

This leads us to our approach where the service consumers first use existing registries to discover the providers that support the interfaces they are interested in. Next, if possible, the consumers query other consumers to help them select an appropriate provider. After each service episode, the consumers score the provider and also rate the other consumers whose recommendations led to the given provider. This rating of consumers acts a feedback mechanism which helps the consumer the next time it needs to select a service provider. We discuss relevant aspects of our approach below.

- *Memory-based.* Memory-based CF keeps a history of user evaluations, and makes predictions based on correlation between users' histories [Breese et al., 1998]. Model-based CF builds a probabilistic model by identifying features that are significant in making a prediction. Our approach is memory-based, but considers each agent's memory separately.
- *Online.* Offline methods consider an entire history of user evaluations to make predictions for the users. By contrast, online methods treat the prediction process as continual and interactive. It involves a learner that corrects itself after each iteration. Thus there is greater personalization of the recommendations.
- *Scoring services.* Explicit voting requires users to provide scores for products along some scale—typically, a small numeric scale such as 1 to 5. Implicit voting involves estimating the user's preferences without the user having to explicitly express his score, by observing patterns such as purchase history, browsing pattern, and the amount of time spent reading an article. Our approach is neutral as to how the scores are obtained, but it assumes that there is an evaluation after the service has been used. Obtaining scores would depend on the user interface and on application-specific details. A service interaction may give an opportunity to obtain an explicit score. We do not consider the user interface aspects of evaluation further in this paper.

2.3 Community-Based Online Service Selection

We now formulate community-based online service selection in generic terms. We model providers and consumers as agents. We refer to the evaluations of service providers by agents as *scores* and the evaluations of agents by other agents as *ratings*. In the following, we refer to an agent as a *user* when the emphasis on an external evaluation such as by a human and as a *rater* when the emphasis is on the act of rating.

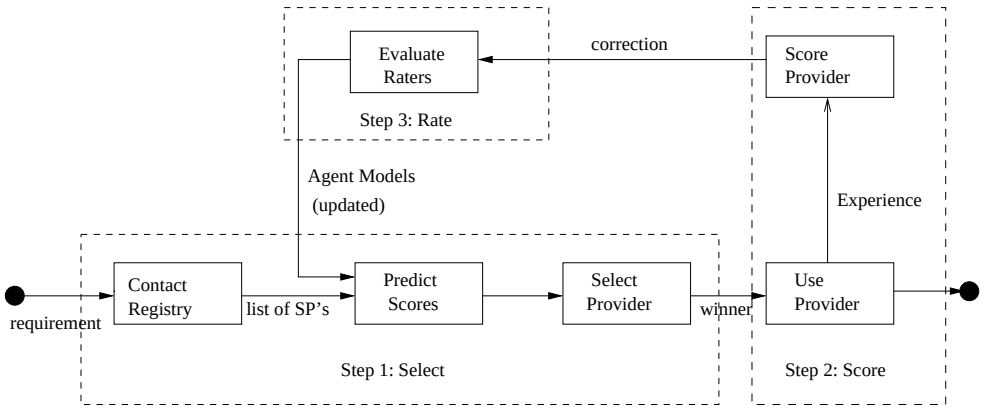

Fig. 2. Community-based online service selection and evaluation

Consumers query other consumers to estimate the quality of service offered by different providers. A query is a list of providers (e.g., as obtained from a UDDI registry) that the consumer needs to select among. A reply is the same list of providers as in the query, but with a score associated with each provider. The score of a provider indicates its evaluation by scoring agent; a null score means the replying agent does not know about the given provider.

Based on the rating of the raters, an agent, representing a consumer, makes a local decision as to which provider to select. Let R_{ik} represent the normalized weight attached by agent N_i to another agent N_k and let S_{kj} be the score given by agent N_k to servicer provider j. Then the score given to provider j by agent N_i can be represented by the following equation, in which N is the total number of agents:

$$S_{ij} = \sum_{k=1}^{N} (R_{ik} \times S_{kj}) \tag{1}$$

Intuitively, each agent may perform a number of tasks. Each task requires the agent to find a provider in a particular *category*. We refer to the agent that is trying to find a provider as the *active agent* and the provider that is selected as the *winning provider*. Figure 2 summarizes our approach, which consists of the following main steps.

- *Select*. Selecting the service provider involves getting a list of service providers rated by other raters and choosing the winner based on a weighted average calculation.
- *Score*. Scoring the selected service during which the provider is engaged and finally rated by the active agent.
- *Rate*. Rating the raters and adjusting the weights that the active agent associates with them.

The active agent goes through these three steps in every iteration. At the end

of each iteration, the active agent evaluates a provider based on its experience and, based on this evaluation, corrects the ratings that it associates with other agents in the community. Instead of associating just one rating with an agent, each agent maintains a list of ratings—one for each service category—for every other agent.

3 Concept-Based Collaborative Evaluation

We propose a concept-based approach that augments community-based online service selection as described above.

3.1 Evaluation Cycle

	a	b	c	d
N_1	0.3	0.5	–	–
N_2	0.2	0.4	0.1	0.5
N_3	–	0.3	0.2	–

Table 1
Agents and the services they have rated

Representation The raters and their scores are stored in a two-dimensional matrix, with the columns representing the different providers, and the rows representing the raters. As an example, let N_1, N_2, and N_3 be three agents who have rated four service providers, a, b, c, and d. Not all providers are rated by all agents. This situation is shown in Table 1. Each agent uses its private matrix to perform local computations, such as rating others.

Choosing the Winner As in memory-based collaborative filtering, to find the winning provider, we consider the deviation from average score measures. In Equation 2, adapted from [Breese et al., 1998], S_{aj} represents the score given by N_a to provider j. $\overline{S_a}$ represents the average score given by N_a to all providers and R_{ai} represents N_a's rating of provider i. Also, in this equation, $N_a \neq i$ and R_{ai} lies in the real interval (0,1).

$$S_{aj} = \overline{S_a} + \frac{\sum_i \left(\left(S_{ij} - \overline{S_i} \right) \times R_{ai} \right)}{\sum_i R_{ai}} \tag{2}$$

Finally, the provider that obtains the highest score (S_{aj}) is chosen.

Evaluating the Service Provider After using the provider, the user can give the true score to the provider by identifying the individual features that are important to him, giving a score to individual features, and finally finding the overall score by calculating the weighted sum of these scores. The particular method that is followed by an individual to come up with a score is not revealed. That is, each agent can preserve its privacy about the features it desires. Moreover, the various agents can employ heterogeneous scoring schemes.

3.2 Applying Concept Lattices

An interesting aspect of our approach arises in the representation and algorithm used by an agent to rate the agents from whom it received evaluations of the providers that it selected. Our representation is based on concept lattices, as defined below [Ganter and Wille, 1999].

Definition 1 A *context* is a triple $\langle G, M, I \rangle$ where G and M are sets and $I \subseteq \langle G \times M \rangle$ is a binary relation between G and M. Here G, M, and I are respectively called the *objects*, *attributes*, and *incidence* of the context. ∎

Definition 2 For $A \subseteq G$ and $B \subseteq M$, if we define: $A' = \{m \in M \mid (\forall g \in A : gIm)\}$ and $B' = \{g \in G \mid (\forall m \in B : gIm)\}$. Then $\langle A, B \rangle$ is a *concept* of $\langle G, M, I \rangle$ if and only if $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$. Here A is called the *extent* and B the *intent* of the concept. The concept $\langle A, B \rangle$ is *nontrivial* if A and B are both nonempty. Concepts are naturally ordered according to set inclusion of their extents. ∎

We model the agents and the service providers as constituting a context. The agents are the objects and the providers are the attributes of this context. The context represents a *has rated* relationship between the agents and the providers. For example, Table 1 represents a context. Figure 3 shows the concepts (C_1 to C_5) in this context. Notice that not all possibilities of intents and extents are concepts in this context. For example, {{N_2},{b, c}} is not a concept in the above context.

When ordered according to their extents, the concepts in a context form a lattice, because the *meet* (greatest lower bound) and *join* (least upper bound) of any two concepts is defined. Such a lattice is called a *concept lattice* [Ganter and Wille, 1999]. Indeed, Figure 3 shows the concept lattice corresponding to Table 1. Here, the edges indicate that the higher concept subsumes the lower concept.

There are three main motivations behind using concepts:

(1) Agents seeing similar facets of the world should be evaluated together.

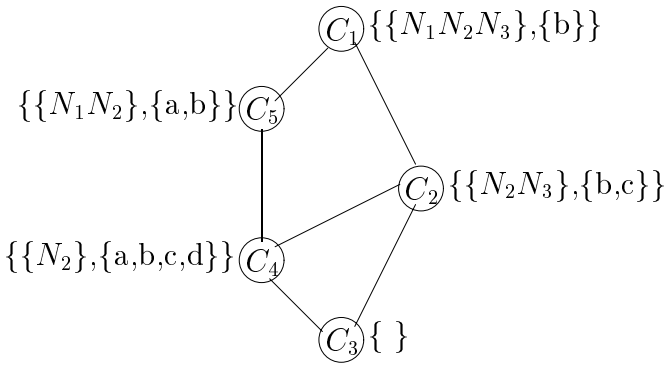

Fig. 3. Concept lattice constructed from Table 1

For example, there is no point in comparing two agents, A and B, where A has evaluated providers $\{a, b\}$ and B has evaluated providers $\{c, d\}$.

(2) All agents are potentially learning about providers. If they have seen only a small number of providers, they should not be penalized or rewarded unduly.

(3) What we are evaluating is the agents' learning capability, so we treat their answer space (providers evaluated) as the sample space their learners are exposed to.

We can apply the principles of concept lattices to analyze the relationships between agents given the scores they gave to the providers. The concept lattice representation has the following properties:

(1) The lattice structure helps visualize the score table, which is just a sparse two dimensional matrix, in a way that is convenient for comparison. This is because, every concept is a complete submatrix of the sparse matrix. The lattice structure gives us a way to choose among such complete submatrices (concepts). The higher the concept in a concept lattice, the greater the number of raters, hence more the number of elements being compared together. Lower the concept, higher the number of rated elements (providers), hence stronger the cause for comparison between the compared elements (raters).

(2) These concept lattices, like the score tables from which they are generated, are decentralized, i.e., one user does not share his lattice with another user. Thus the decision of keeping a memory of the concept lattice generated during every iteration, i.e., every time a score table is generated as a result of querying neighbors for service providers, is left to the individual user or agent. In our simulations, we destroy the lattice after every iteration.

(3) The complexity of the above computation over concept lattices is $O\left(\left|B\left(N, P, S\right)\right| \times |N|^2 \times |P|\right)$, where N is the number of agents, P is the number of providers, S is the number of scorings, and $B\left(N, P, S\right)$

is the set of all concepts so generated. The algorithm is borrowed from Ganter and Wille [1999].

Evaluating the Raters We associate a confidence value with every rating. We decompose the rating that an agent gives to another agent into two values—the *supposed rating*, r_{ai}, and the *confidence* in the rating, q_{ai}. We compute the effective rating R_{ai} as follows:

$$R_{ai} = (r_{ai} + 1)^{q_{ai}} - 1 \tag{3}$$

Equation 3, based on a formula given in [Chen and Singh, 2001], has some desirable properties. Specifically, R increases with r and q when the other variable is fixed; it yields $R = 0$ when either r or q is 0; it yields $R = 1$ only when both r and q are 1; and finally $R = r$ when q equals 1. Also, the rating, R, grows quicker with q than with r. This is a desirable feature since the emphasis of rating should be on the confidence in rating rather than on the actual value of the rating itself.

Traversing the Concept Lattice In a concept lattice, a parent concept is a superset of a child concept. That is, higher concepts contain more agents than lower concepts. Thus the number of agents considered for evaluation is maximized by choosing concepts that are closer to the root. But, in order to compare agents (thus evaluate them), we must have a standard of comparison. This prevents us from choosing arbitrary concepts for evaluation. The concept containing the active agent and the winning provider is an obvious starting point, since we can directly compare the scores given by other agents with scores given by the active agent. We can then proceed to other concepts (to evaluate other agents) and make indirect evaluations by choosing concepts in an order that will guarantee that at least one standard of comparison is present in the concept (e.g., a previously evaluated agent). Algorithm 1 takes these factors into consideration.

Algorithm 1 Evaluate-Agent()

1: Search concept lattice to find a the maximal concept, say C, containing the winning provider.
2: Evaluate all the raters in C.
3: Let D be a maximal nontrivial concept subsuming C. Evaluate all raters in D.
4: Traverse the lattice from the top down, visiting nontrivial concepts until all raters in the context have been evaluated.

Since all the agents would have been reevaluated when Algorithm 1 completes, it is unnecessary to consider any other concept that has not been handled. For

this reason, such concepts are ignored.

Example 1 This example shows how Algorithm 1 executes on the concept lattice of Figure 3. Suppose that the winning provider was found only in concept $[N_2]$ (e.g., if provider d was the winning provider). Below, we show how the above algorithm is run on the lattice:

(1) Concept C_4 is found via a breadth first search to find a concept that contains the winning provider.
(2) Agent N_2 is evaluated during the course of handling concept C_4.
(3) Concept C_5 is the maximal nontrivial concept subsuming concept C_4, so it is handled next. This results in agent N_1 being included in the set of agents that have been evaluated.
(4) Since all the agents have not yet been evaluated (Agent N_3 is still left out), the lattice is traversed top down. This would result in concept C_2 being handled and agent N_3 being evaluated.

Now all agents have been evaluated and hence the algorithm terminates. ▮

3.3 Rating Raters via Concepts

When agents collaborate to select service providers a potential risk is that they may have widely differing preferences regarding service providers. Thus a high score by one agent may not correspond to a high score by another. Therefore, a key aspect of our approach is that agents can rate each other to effectively learn about how much credence to give to each other's scoring of a provider. Informally, this rating corresponds to the correlation between the scores given to providers by the various agents. However, there are some complications, which we must accommodate. One, an agent scores a provider only if it was chosen as a winning provider: thus the data is initially quite sparse. Two, the agents may not always have scored the same providers, so we need to propagate their ratings through concepts.

The important step during evaluating raters is the way in which concepts are handled. The end result is that the rating (r) and confidence (q) of the agents that are present in the concept are altered to better represent the real world.

	P_1	P_2	P_3	r_{t-1}	q_{t-1}	r_t	q_t
N_1	x_1	y_1	z_1	r_1	q_1	?	?
N_2	x_2	y_2	z_2	r_2	q_2	?	?

Table 2
Handling concepts

Table 2 shows the scenario for a concept with two raters (N_1, N_2) and three rated services (P_1, P_2, and P_3). Further, in order to facilitate the calculation of r_t and q_t, we could divide this into three subscenarios as described in the following subsections. Recall that the *active agent* refers to the agent under consideration (who is seeking a service provider). And, *winning service provider* refers to the service provider that is finally selected from among a previously discovered list of providers.

3.3.1 Scenario 1: Early Stages

During the first few iterations, the active agent would not have rated enough service providers to have a row-wise comparison with other users. The scenario is depicted below:

	P_1	P_2	P_3	r_{t-1}	q_{t-1}	r_t	q_t
N_1	x_1	y_1	z_1	r_1	q_1	?	?
N_2	x_2	y_2	z_2	r_2	q_2	?	?
active-user	x_3	–	–	–	–	–	–

In this scenario, we have a concept that contains a service provider (1) that has been evaluated by the active user.

Updating the Ratings The new rating (r_t) depends on the following:

(1) The current rating (r_{t-1}).
(2) The absolute difference between the scores given to the service provider that has been evaluated by the rater and the active user.

$$d = |s_{kj} - s_{aj}| \tag{4}$$

We use the following equation to rate agents in this scenario:

$$r_t = r_{t-1} + \rho \left(\frac{\beta_t}{1 + d} \right) \tag{5}$$

In Equation 5, ρ is 1 if d is less than a *threshold*, or else, it is -1. The damping factor β_t is defined below.

Normalizing The difference d in Equation 5 is normalized to be in the real interval $(0, 1)$. In order to prevent agent ratings from growing very large,

we normalize the rating such that $\sum r_{t-1} = \sum r_t$. This could serve as a minor incentive for the agents to reply to queries, since the only way they can increase their rating is by competing with others (they cannot just wait for others' ratings to go down).

Damping We introduce the damping factor, β_t, in Equation 5 in order to damp the increase and decrease of the rating. This damping factor should reduce the incremental value ($1/(1+d)$ in Equation 5) at the extremities, i.e., the reduction of an already low rating, and the increase of an already high rating should be low. We use Equation 6, in which λ is a positive constant and r'_{t-1} is the minimum of r_{t-1} and $(1.0 - r_{t-1})$.

$$\beta_t = \lambda r'_{t-1} \tag{6}$$

3.3.2 Scenario 2: Later Stages

This scenario is same as in Section 3.3.1 except that the active agent has now rated enough service providers to be able to make a row-wise comparison with other agents. The considerations for this scenario is same as for the previous one, except that the vector similarity between the active user and the other user replaces the additive factor in Equation 5.

Below, we represents the set of agents in the community, and J represents the set of providers. Breese et al. compare several algorithms to calculate the correlation between vectors [1998]. Pearson's r-correlation with some extensions performs well. Accordingly, user i's rating of another user k is given by:

$$r_{ik} = \frac{\sum_j (s_{ij} - \overline{s_i})(s_{kj} - \overline{s_k})}{\sqrt{\sum_j (s_{ij} - \overline{s_i})^2 \sum_j (s_{kj} - \overline{s_k})^2}} \tag{7}$$

Here $\{j \in J\}$ is the set of all service providers, s_{ij} refers to the score that user i has given to the provider j, and $\overline{s_i}$ refers to the average score given by user i. Note that r_{ik} is positive or negative depending on the degree of correlation. The ratings are normalized as in Scenario 1.

3.3.3 Scenario 3: Propagating Evaluations Through Concept Chains

An agent Z that is considered for evaluation satisfies at least one of the following conditions:

(1) Z is in a concept that contains the winning provider in its intent.

(2) Z is in a concept that has agents that have been evaluated by virtue of satisfying condition 1.
(3) Z is in a concept that has agents that have been evaluated by virtue of satisfying conditions 1 or 2 above. (Note that this is the generic case of condition 2.)
(4) Z is in a lonely concept that cannot be reached through any of the above ways. Such concepts are not handled (i.e., the agents belonging to such concepts are not evaluated).

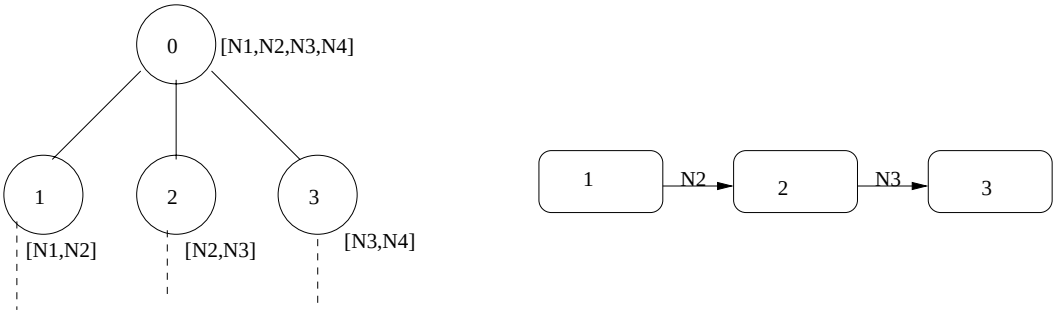

Fig. 4. Propagating ratings: lattice fragment (l) and propagation chain (r)

In order to facilitate indirect evaluation of raters (to find friends of friends), we need a mechanism for propagating ratings and confidence through concepts. Consider the concept lattice fragment in Figure 4(l). In this lattice, if concept 1 contained the active agent, N_1 (making a direct evaluation of the agents in this concept possible), then the ratings (and confidence) would have to propagate to concept 2 through N_2, and to concept 3 through N_3, via concept 2. The propagation graph is shown in Figure 4(r).

	P_1	P_2	P_3	r_{t-1}	q_{t-1}	r_t	q_t
N_1	x_1	y_1	z_1	r_1	q_1	?	?
N_2	x_2	y_2	z_2	r_2	q_2	?	?
N_3	x_3	y_3	z_2	r_3	q_3	r'_3	q'_3

If Z satisfies condition 3 and not conditions 1 and 2, we have a scenario in which the concept being handled has at least one rater that has already been evaluated, perhaps in a previously handled concept. We have to somehow find a way of propagating this new rating and confidence to the other agents in this concept. One such scenario, with the propagating agent being N_3, is shown in the table above.

Updating the Ratings Ideally the rating, r, given to an agent is a vector with the attributes for rating representing the dimensions of this vector. Thus a new rating, r_t, in an indirect evaluation, depends on the following:

(1) The current rating (r_{t-1}).
(2) The vector distance of the rater's rating and the linking rater's rating ($d2$).
(3) The vector distance between the active agent and linking agent ($d1$).
(4) The distance between the previous and current rating of the linking rater (l).

A graphical representation of the situation is shown in Figure 5.

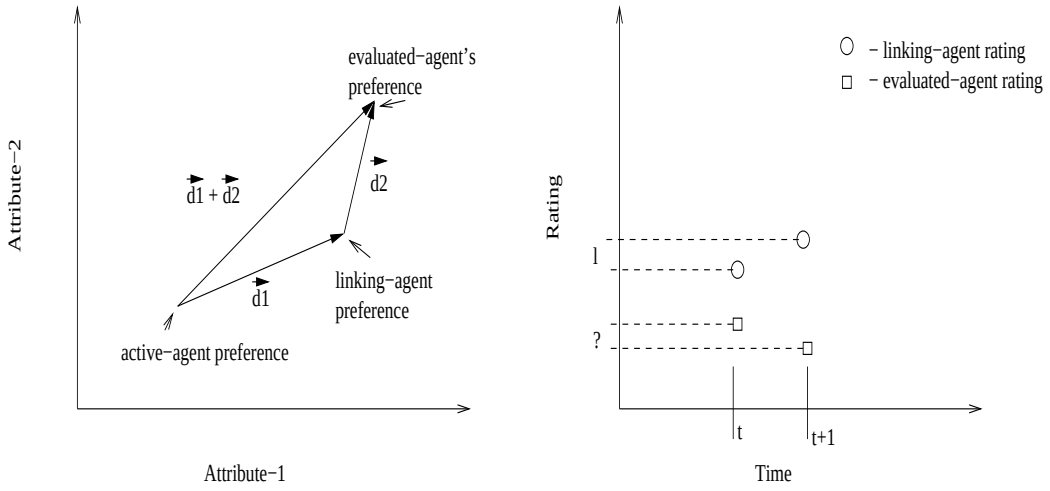

Fig. 5. Method for propagating ratings through concepts

Since we use a scalar value to represent ratings, the distance between ratings is also a scalar. So, we use Equation 8 as an approximation to the ideal case, in order to calculate the new rating r_{t-1}. Here, β_t is the damping factor calculated as in Equation 6.

$$r_t = r_{t-1} + \beta_t \left(\frac{d1 + d2}{d1} \right) l \tag{8}$$

3.3.4 Updating the Confidence

We use a generic formula to update the confidence in rating (q in Equation 3) in all the scenarios discussed in Section 3.3. In general, the confidence in rating must go up after every evaluation of the agent, irrespective of whether or not the rating of the agent itself was increased. The case when confidence goes down (e.g., when an agent behaves erratically) could be a possibility in a real-life scenario. We defer exploring this topic to future work.

The requirement for the value of confidence is summarized below:

(1) The value of confidence is kept in the real interval (0,1).
(2) Confidence is initialized to a low value and is incremented every time there is an opportunity to evaluate the agent.

(3) The amount by which the confidence is increased during every iteration depends on the degree of indirection while evaluating the agent's rating, i.e., the increment decreases with the propagation length of the ratings. For example, in Figure 4, the amount by which the confidence in the rating of N_4 is increased should be less than the amount by which the confidence in the rating of N_3 is increased, which in turn should be less than the amount by which the confidence in the rating of N_2 is increased.

With this motivation, the following equation is used. Here, α is a positive scaling constant and δ is a damping factor. The damping factor is set to the reciprocal of the sum of the confidences associated with every category of the agent being evaluated, i.e., $\delta = 1/\sum_c q_{tc}$.

$$q_t = \alpha \times \left(q_{t-1} + \frac{\delta}{(l+1)} \right) \tag{9}$$

4 Experimental Validation

During our experiments, we iterate through a list of *tasks*, each of which specifies an active agent and a service category. The tasks are randomly chosen, but legal. Multiple active agents make predictions for providers in multiple categories. For each task, the agent attempts to select a service provider matching the given category. If it does, it selects a winning provider along with a predicted score for it—the predicted rank for the winning provider is 1 (that's why it wins). Next, the score given by the user to this service provider is revealed and used to assign the actual score for this provider and to update the ratings given by the active agent to other agents.

Each agent maintains an *observation matrix*, which captures the scores that it knows of that were given by consumers to various providers. As the experiment proceeds, this matrix would grow in both dimensions. At any given time, the observation matrix is subsumed by the *truth matrix*, which represents all the ratings available through the dataset being used. Our experiments ensure that the information about users and the scores they give to various providers is not shared except to the extent allowed by the above algorithm.

4.1 Datasets for Service Selection

No extensive real datasets of scorings of open services are available. Therefore, we consider two means of constructing datasets to evaluate service selection approaches. First, we develop an artificial dataset, which includes users

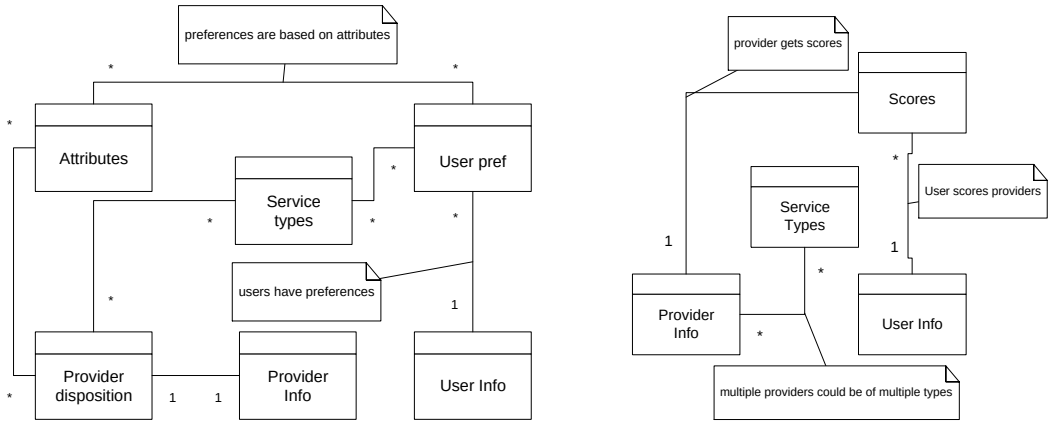

Fig. 6. Conceptual schemas for our artificial dataset (l) and MovieLens (r)

along with their associated feature preferences (what they need), and service providers along with their associated feature values (what they provide). Figure 6(l) provides a conceptual schema for this dataset. This is a simplified version of the conceptual model proposed in [Maximilien and Singh, 2002]. Based on these preferences and feature values, the scores given by users to service providers can be programmatically calculated. This is no more than a convenience for the simulation—in real-life, the users would know their scores. In practice, the attributes would be supplied and services would be appropriately marked up through a semantic Web service approach. This is heartening, because as semantic Web services become prevalent, they will be able to facilitate selection based on how agents rate other agents, not just through filtering by a repository for the given description.

The artificial dataset shows how to exploit the semantics of service markups. However, because the artificial dataset avoids the vagaries of scoring service providers, we also consider another dataset which involves scores supplied by real users. This dataset is an adaptation of the well-known MovieLens dataset [MovieLens, 2002]. MovieLens contains 100,000 scores assigned to 1,682 movies by 943 users. The scores are in the range 0 to 5. Figure 6(r) describes MovieLens' simple conceptual schema, which including users, movies, and scores. It is naturally a product database where the movies are the products. However, we interpret movie genres as service categories and individual movies are implementations of their genre. The scores provided by the users to various movies are used in our reasoning. Besides the genre, attributes of movies are not used. This demonstrates that the proposed approach can work even when there is little or no semantic markup.

Thus, an experiment might ask agent N_1 to select a service (movie) of the category (genre) *action*. N_1 may finally come up with *Rambo* along with a predicted score—to be selected, Rambo would have the highest score predicted by N_1. Now, using the dataset, we would determine the true score and ranking of Rambo as well as the score of the actual best movie for N_1. This would

enable us to measure N_1's accuracy.

Further, to ensure that the plots we obtained were not arbitrary, we partitioned MovieLens into three independent datasets. The reported results were stable across these three datasets.

4.2 Previous Algorithms Adapted for Service Selection

Our approach is compared with two previous approaches adapted for service selection. These approaches can predict a winning provider and evaluate the raters, thus providing alternative means of carrying out the functionality depicted by the first and the last blocks of Figure 2.

Correlation In this approach, an agent's rating of another agent is based on the Pearson correlation between the score of the two users (Equation 7). While making a prediction, a weighted average of the scores given by the agents (weighted by their correlation with the active agent) is computed for each competing service provider. The provider with the highest score is declared as the winner.

Here, all the scores given by the two users until now are considered. By contrast, in our approach, correlation is used only to find the similarity in scores given by two users in the current iteration.

Generalized Learn Relationship (GLearn) GLearn is a generalized version of the weighted majority algorithm (described in Section 2.1), where instead of just a binary vote, users vote for all values within a permitted tolerance of their true value [Nakamura and Abe, 1998]. Specifically, let A denote the range of values in the truth matrix M. Also, for any $a \in A$, let $V(a)$ denote the set of prediction values that are permissible when the correct value is a. Then, a prediction is made as follows:

$$\hat{M}_{ij} = \begin{cases} \arg\max_{a \in A} \sum_{k:O_{kj} \in V(a)} w_{ik} \text{ if } (\{k : O_{kj} \neq *\} \neq \phi) \\ C_0(\text{a constant}) \qquad \text{otherwise} \end{cases} \tag{10}$$

After obtaining the true value (the second step in the three-step approach described in Section 3.1), the weights are updated as follows:

$$w_{ik} = \begin{cases} (2-\gamma)w_{ik} & \text{if } O_{kj} \in V(M_{ij}) \\ \gamma w_{ik} & \text{if } O_{kj} \notin V(M_{ij}) \end{cases} \tag{11}$$

4.3 Results

We present some technical results comparing the above two approaches *Correlation* and *GLearn* with *Concept* (our approach).

4.3.1 Learning Curve

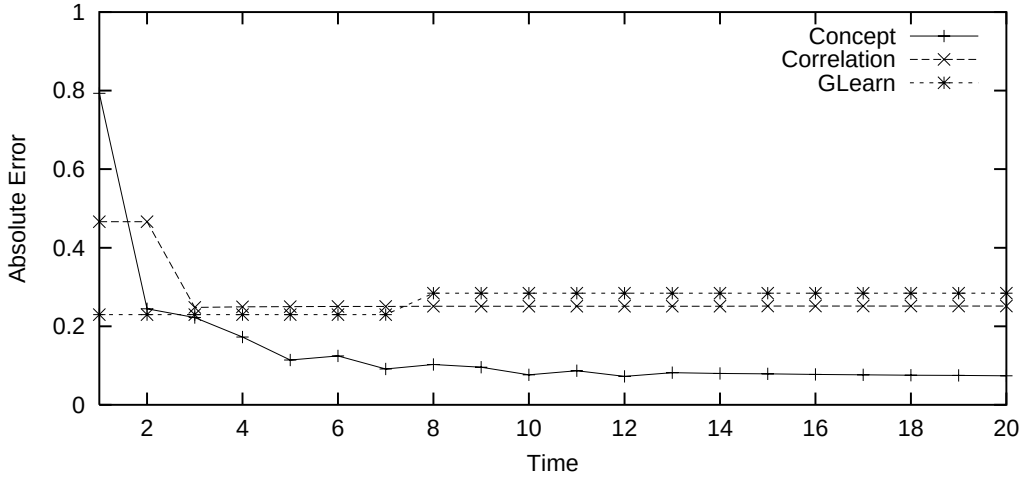

Fig. 7. Comparison of learning curves, i.e., the time taken to reach steady state

We consider the time taken to reach the steady state, after which prediction accuracy remains almost constant. Initially, the agents associate a default rating (which depends on the specific approach) with each other. Since inconsistent scorings can confound the results on the quality of the algorithm, we use the artificial dataset for this experiment. Figure 7 shows that Concept converges slightly faster to the steady state than the other two approaches. And, for the same dataset, the absolute error made in Concept is lower than the error in the other two approaches.

4.3.2 Steady State Accuracy

We compare the accuracy of the three approaches in their steady state by recording the errors made during predictions (Figure 8). Here all active agents

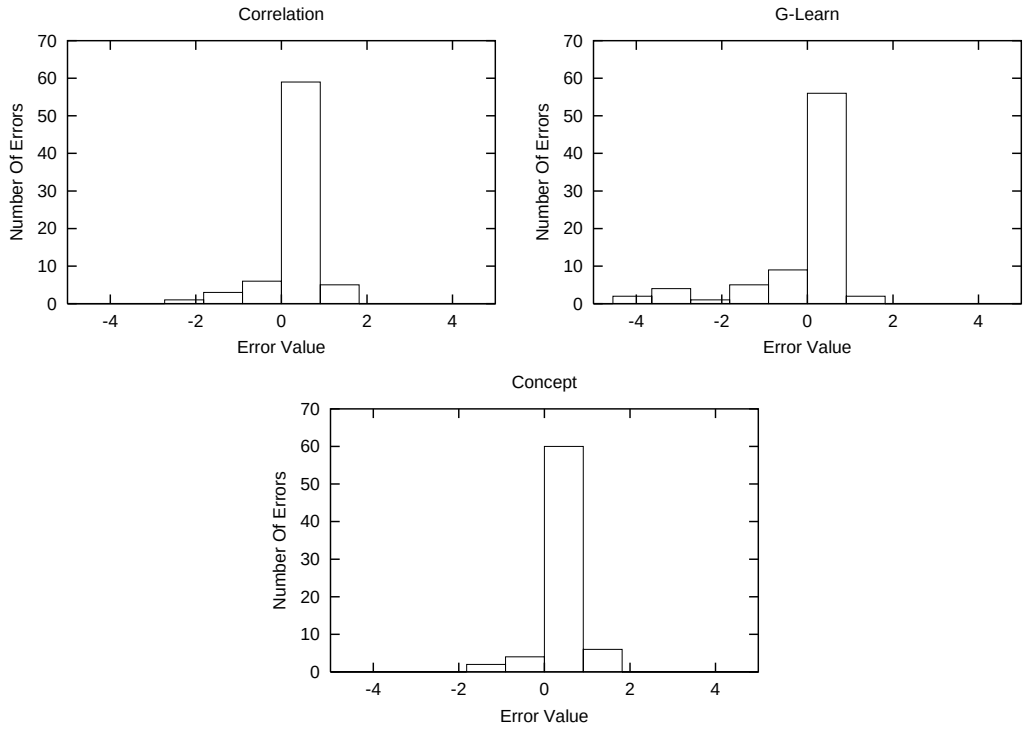

Fig. 8. Steady state accuracy test for MovieLens

are initialized with 30% of the truth matrix. From Figure 8 we see that Correlation and Concept yield similar error distributions and have slightly more perfect predictions (shown as an error of zero) than GLearn. More importantly, Correlation and Concept yield few predictions with a large error margin (> 2), whereas GLearn makes several such predictions.

4.3.3 Average absolute error comparison

We log the predicted score for the winning provider and compute the absolute error in prediction based on its actual score. We measure the average absolute error values of the predictors, after initializing them with a set of tasks. If N denotes the number of tasks to be completed, we initialize the predictors with a certain percent of these tasks, say X, and then measure average absolute error values for the remaining $(100-X)$ percent of the tasks. The performance of the three predictors in this measurement is shown in Figure 9. N was chosen to be 500 in our experiments. The errors trend downwards with Concept performing slightly better than the others at most points.

4.3.4 Ordinal Error

The steady state accuracy test records the error of prediction irrespective of the quality of prediction. That is, we measure the error in the rank of the

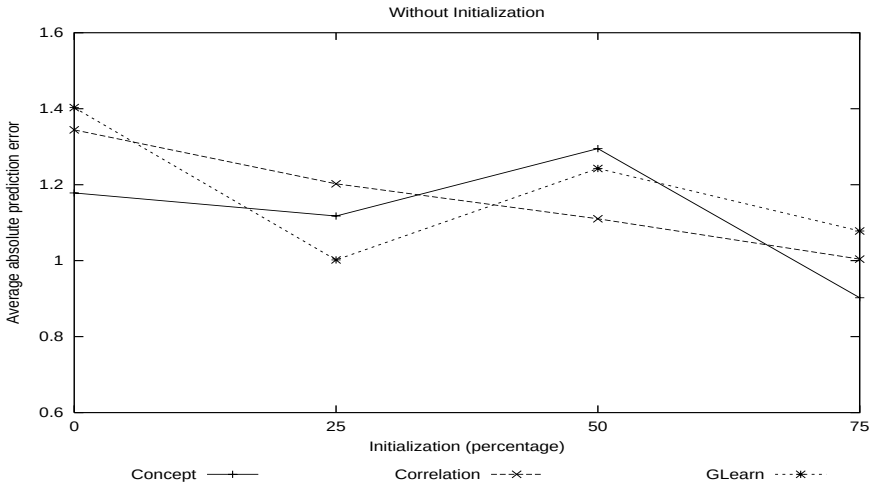

Fig. 9. Variation of absolute error with initialization for MovieLens

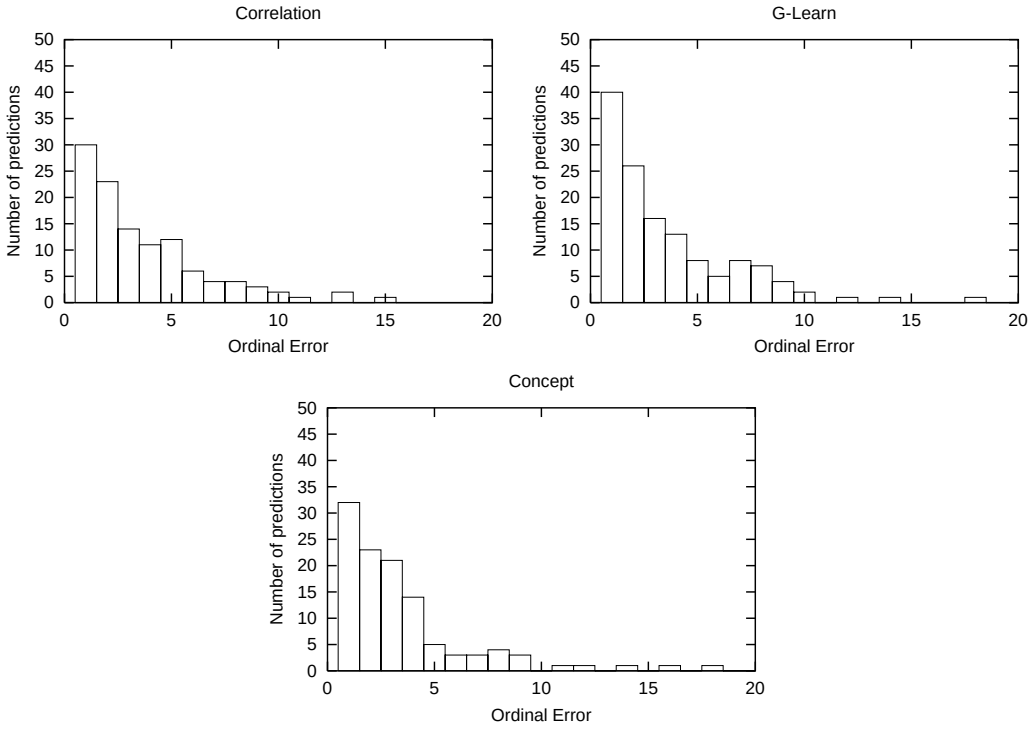

Fig. 10. Ordinal error test for MovieLens

winning provider, which is predicted to have a rank of 1. Therefore, we define the *ordinal error* as the true rank of the winning provider among the providers considered in the current iteration. The true rank is obtained from the truth matrix.

The experimental setup is the same as for steady state accuracy. Figure 10

shows plots of the ordinal errors made in the three approaches. GLearn yields fewer total errors than Concept, which in turn is better than Correlation. However, the mean ordinal errors for Concept, GLearn, and Correlation are 3.4, 3.6, and 3.7, respectively. That is, Concept and GLearn perform well in this test.

4.3.5 Spearman's Correlation

We extend the previous experiment and calculate the overall accuracy of ordering of the service providers by the various predictors. We calculate Spearman's correlation (Equation 12) between the predicted order of the providers and the actual order computed from the truth matrix. Spearman's correlation of two lists (C and D in Equation 12) is a value between -1 (opposite order) and 1 (same order).

$$c = 1 - \frac{6 \sum (C_i - D_i)^2}{n\left(n^2 - 1\right)} \tag{12}$$

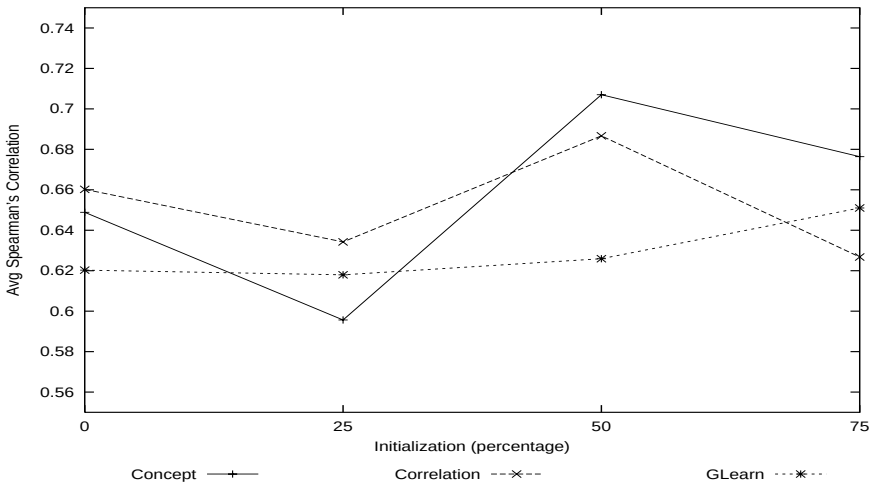

Fig. 11. Calculation of average Spearman's correlation for MovieLens after varying initializations

We carried out experiments to calculate average value of Spearman's correlation of predictions for each predictor after initializing them with tasks. The results are summarized in Figure 11. Again N, the total number of tasks, was chosen to be 500. The trend is upwards with Concept producing a higher Spearman correlation with increasing initialization.

5 Discussion

We now summarize our contributions and discuss some important directions for further research.

5.1 Conclusions

As Web services become established, semantic approaches for modeling and deploying them are emerging. In particular, the problem of service selection is beginning to attract attention from researchers. We develop an approach that enables agents to collaborate to help each other select the best services. The approach can work with rich service quality attributes where they are available but can also work where the agents merely interact with one other and empirically determine how much to rate each other's evaluations of service providers.

Services differ from products significantly in terms of how they are discovered, selected, delivered, and evaluated. In particular, services call for a distributed approach that preserves the autonomy of the various parties. Our approach takes these special properties of services into account to yield a credible means for service consumers to share information to help each other make refined selections. Community-based approaches face the challenge of bootstrapping new users. Our approach accommodates new users by beginning to make predictions with little data and then revising the scores and ratings through experience.

Our approach for selecting service providers borrows interesting elements from conventional approaches. Like collaborative filtering, it uses correlations though not centrally; like reputation systems, it records ratings though not centrally; like P2P systems, it includes neighbors that an agent contacts; like referral systems, it employs ratings that an agent revises through experience along the lines of the weighted majority algorithm. In addition, it introduces a novel way for comparing raters borrowing concepts from lattice theory.

5.2 Directions

Ideally, a service consumer may select a provider based on a combination of features and the quality of service provided. In principle, a score could be computed based on the weights attached to the various features. But a consumer may know neither the features nor the weights associated with them. By contrast, in a community-based approach, consumers can share their scorings

of providers. However, each consumer must be able to rate those who have scored the providers. Our approach can be extended to accommodate deeper representations of the conceptual models of attributes used for ratings and reputations as well as means to dynamically discover attributes using which ratings of services can be compared and desirable services found [Maximilien and Singh, 2002].

In our approach, the confidence attached to a rating goes up after every evaluation of the agent, irrespective of whether or not the rating itself is going up. But in real life the confidence could decrease due to ambiguity of evaluation or erratic behavior. However, we have not explored this direction since we do not consider the case of varying interests of raters. A probabilistic approach for trust, e.g., [Barber and Kim, 2001; Yu and Singh, 2002], could be applied in this case.

Another class of efforts is motivated from structuring and composing services. The Business Process Execution Language (BPEL) describes compositions of services as workflows [BPEL, 2002]. McIlraith et al. apply planning techniques to compose services [2001]. It would be interesting to relate the selection of services to the workflows or plans in which they are intended to be embedded.

References

Barber, K. S., Kim, J., 2001. Belief revision process based on trust: Agents evaluating reputation of information sources. In: Falcone, R., Singh, M. P., Tan, Y.-H. (Eds.), Trust in Cyber-societies. Vol. 2246 of LNAI. Springer-Verlag, pp. 73–82.

Berners-Lee, T., Hendler, J., Lassila, O., 2001. The semantic Web. Scientific American 284 (5), 34–43.

BPEL, Jul. 2002. Business process execution language for web services, version 1.0. Www-106.ibm.com/developerworks/webservices/library/ws-bpel.

Breese, J. S., Heckerman, D., Kadie, C., 1998. Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence. pp. 43–52.

Chen, M., Singh, J. P., 2001. Computing and using reputations for Internet ratings. In: Proceedings of the 3rd ACM Conference on Electronic Commerce. ACM Press, pp. 154–162.

DAML-S, Jul. 2002. DAML-S: Web service description for the semantic Web. In: Proceedings of the 1st International Semantic Web Conference (ISWC). Authored by the DAML Services Coalition, which consists of (alphabetically) Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry R. Payne and Katia Sycara.

Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., Harshman,

R., 1988. Using latent semantic analysis to improve access to textual information. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems. ACM Press, pp. 281–285.

Epinions, 2002. Home page. Http://www.epinions.com.

Ganter, B., Wille, R., 1999. Formal Concept Analysis. Springer, Berlin.

Hendler, J., McGuinness, D. L., 2001. DARPA agent markup language. IEEE Intelligent Systems 15 (6), 72–73.

Kan, G., 2001. Gnutella. In: Oram [2001]. Ch. 8, pp. 94–122.

Littlestone, N., Warmuth, M. K., 1994. The weighted majority algorithm. Information and Computation 108 (2), 212–261.

Maximilien, E. M., Singh, M. P., Dec. 2002. Conceptual model of Web service reputation. ACM SIGMOD Record 31 (4).

McIlraith, S. A., Son, T. C., Zeng, H., Mar. 2001. Semantic Web services. IEEE Intelligent Systems 16 (2), 46–53.

MovieLens, 2002. Home page. Http://movielens.umn.edu.

Nakamura, A., Abe, N., 1998. Collaborative filtering using weighted majority prediction algorithms. In: Proceedings of the 15th International Conference on Machine Learning. Morgan Kaufman, pp. 395–403.

Oram, A. (Ed.), 2001. Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology. O'Reilly & Associates, Sebastopol, CA.

Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J., 2000. Analysis of recommendation algorithms for e-commerce. In: ACM Conference on Electronic Commerce. pp. 158–167.

Singh, M. P., Jun. 2002. The pragmatic Web. IEEE Internet Computing 6 (3), 4–5, instance of the column *Being Interactive*.

Singh, M. P., Yu, B., Venkatraman, M., Apr. 2001. Community-based service location. Communications of the ACM 44 (4), 49–54.

Trastour, D., Bartolini, C., Gonzalez-Castillo, J., Jul. 2001. A semantic Web approach to service description for matchmaking of services. In: Proceedings of the International Semantic Web Working Symposium (SWWS).

UDDI, 2002. Universal Description Discovery and Integration. Http://www.uddi.org.

WSDL, 2002. Web Services Description Language. Http://www.w3.org/TR/wsdl.

Yu, B., Singh, M. P., Nov. 2002. Distributed reputation management for electronic commerce. Computational Intelligence 18 (4), 535–549.

Zacharia, G., Moukas, A., Maes, P., 1999. Collaborative reputation mechanisms in electronic marketplaces. In: Proceedings of the 32nd Hawaii International Conference on System Sciences Minitrack on Electronic Commerce Technology.