A User Centered Matchmaking and Ranking System

Rehab Duwairi and Ahmad Rawashdeh
Faculty of Computer and Information Technology
Jordan University of Science and Technology
rehab@just.edu.jo, ahmadyr2003@yahoo.com

Abstract
This paper provides a flexible model for matchmaking. It is based on supporting the service and demand suppliers with the necessary tools for selecting the subset of attributes that describe a service or demand. The model also allows the users to adjust existing attributes or add new ones. Finally, it allows the users to adjust the weights of the attributes. Automatic ranking was also supported; the set of matches is returned sorted in decreasing order of their closeness to the demand. A prototype was implemented to test the viability of the model and we chose the domain of job matching. The initial results show that the model has a potential success.

Keywords: Matchmaking, ranking, distance computation, dynamic rank calculation, flexible demand-service specification.

1. Introduction
Matchmaking is the process of finding the appropriate supplies for a demand, service for a request [2]; it has wide applications, such as dating systems, job matching systems, hotel reservations, …, etc. The standard model for matchmaking is based on mapping services to requests; where both services and requests are represented based on predefined attributes set by the system designer or service/demand provider. Typically, the demander creates the required service description by entering values or by selecting items from a predefined set of attributes displayed in a form. Alternatively, s/he may define the required service description by writing its description using natural language. Lastly, s/he may create software agents, if the matchmaking system supports multi-agents.

In agent-based matchmaking, communications between agents could follow the facilitator (Middle agent) model, where there is one or more central directors that help the demander locate the service or task. Agent communications, on the other hand, might follow the market model, where bids and offers are broadcasted to all of the agents in the market [4, 5]. Some researches study the model of peer-to-peer matchmaking and clustering [2, 7] and compare the performance of the latter by the previously mentioned models [4]. The work reported in [2, 3] discusses semantic matchmaking. Constantinescu, Binder and Faltings create a new query language DirQL for matching and ranking in service directories, where they provide an efficient search through the directory [1]; and also a framework for matchmaking was developed in [6] with the intention of achieving openness and flexibility in matchmaking.

The work reported in this paper is concerned with what to query rather than how to query or communicate and locate the desired provider. In particular, we are concerned with the limitations in defining the demand/service attributes, and if the demander can query the values entered by a supplier related to attributes defined by the demander himself.
We believe that the process of matchmaking, in its current state, is less flexible than what it should be - the attributes of the service description (advertisement, demand) are limited, fixed and pre-determined by the designer of the system. This makes the system applicable only for specific scenarios. The authors suggest a model where each demander has the choice or ability of defining service attributes and specifying their weights. Also, the researchers discuss issues that might arise as a consequence of such flexibility and suggest some possible solutions. Lastly, they explain how the process of ranking the matches (hits) could be achieved.

This paper is organized as follows: Section 1 presents an introduction to matchmaking. Section 2 illustrates the design philosophy. Section 3, by comparison, demonstrates the prototype that we have developed to implement our design philosophy. Finally, conclusions of this work are described in Section 4.

2. Design Philosophy

We believe that flexibility is important in the process of matchmaking. The user should be able to customize and configure the process to suit her/his needs. Therefore, the more control the user has on the process, the more the result of matching will reflect her/his needs. Even if an autonomous matching is provided by a software agent, user customization and control should not be ignored, because the absence of user control could sometimes lead to unsatisfying results.

The following subsections describe the design choices that the authors have taken to ensure user involvement in the matching making process.

2.1 The Proposed Model

The proposed model consists of two features that both can be used or either one to enhance the matchmaking process. The first allows the user to define a ranking strategy and the second provides the user the ability to define his service description by creating new attributes or by using existing ones. The model is centered on two entities, namely: the demander and the supplier. The demander may define the service description attributes from scratch or may use existing attributes that have been used by other demanders with similar interests or may select a combination of both choices.

2.2 Weights of Attributes

Each demander has the ability to define the points (weights) for each attribute value. The values with no points are dismissed from the matching query. This allows the demander to query for multiple values of the attribute instead of one value as long as to determine the ranking strategy. Also the weights for each attribute value could be stored in the system.

The user is given the choice to define the ranking strategy by entering weights for each attribute value, or by using a direct match. If s/he chooses to enter weights, a textbox appears beside each attribute value, where the match is done as in the usual method, but with the exception that more than one attribute value could be selected. The most important difference, between the suggested method and other methods, is that these weights will be used to rank the result. A function could be used to calculate the overall weight of each row in the result of a given query. Note that a row in the result represents an entity that matches a request (query). For example, suppose that in a job matchmaking system the user has three attributes in his service description which are programming languages displayed as a collection of checkboxes, university
degree and country as dropdown list. Assume that one of the demanders (Employers) makes this request:

Get All Job Seekers
((C# {3}, Vb.Net {4}) , (Bachelor{2}, Master{4}) , (Jordan{3}, USA{2}, Russia{1}))

Where the request format is:

( attribute_1-value_1 {weight},…….attribute_i-value_i {weight}),
( attribute_2-value_1 {weight},…….attribute_j-value_j {weight}),……
( attribute_k-value_1 {weight},…….attribute_k-value_k {weight})

Then the result should contain all of the suppliers (job seekers) who have either C# or VB.net as one of their programming languages. Bachelor or Master as their University Degree; and Jordan, USA or Russia as their Country, sorted in descending order according to their rank which is calculated from their respective weights. The rank of a matching entity is calculated by dividing the sum of the seeker points by the maximum number of points that could be achieved (the ideal scenario); 14 in this example. Table 1 shows a sample of how the result to the above query might look like:

<table>
<thead>
<tr>
<th>Seeker Name</th>
<th>Programming Languages</th>
<th>University Degree</th>
<th>Country</th>
<th>Email</th>
<th>Weight</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simon</td>
<td>Java, C#, Vb.net, Delphi</td>
<td>Master</td>
<td>USA</td>
<td><a href="mailto:Simon@yahoo.com">Simon@yahoo.com</a></td>
<td>13/14</td>
<td>92.85%</td>
</tr>
<tr>
<td></td>
<td>3+4 = 7</td>
<td></td>
<td>4</td>
<td>2</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Smith</td>
<td>C++, Vb.net</td>
<td>Master</td>
<td>USA</td>
<td><a href="mailto:Smith@yahoo.com">Smith@yahoo.com</a></td>
<td>10/14</td>
<td>71.42%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>John</td>
<td>Pascal, Vb.net</td>
<td>Master</td>
<td>Russia</td>
<td><a href="mailto:John@yahoo.com">John@yahoo.com</a></td>
<td>9/14</td>
<td>64.28%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td>1</td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Richard</td>
<td>C#, J#</td>
<td>Bachelor</td>
<td>Russia</td>
<td><a href="mailto:Richard@gm.com">Richard@gm.com</a></td>
<td>6/14</td>
<td>42.85%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td>2</td>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Ivan</td>
<td>Java, Fortran</td>
<td>Master</td>
<td>Russia</td>
<td><a href="mailto:Ivan@gmail.com">Ivan@gmail.com</a></td>
<td>5/14</td>
<td>35.71%</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td>4</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Total Weight and Rank Calculation

2.3 Dynamic Attributes

Using a fixed template by all users to describe their services may be sometimes inapplicable and reduces the flexibility of the system, especially if the system is modeled from a real world process that is usually done in different methodologies by different people.

One of these processes for example is job matchmaking along with its many various methods and applications, where the form used to apply for a teacher job may be different from the one used by companies seeking a driver. From our point of view, this leads to the need for a dynamic attributes model.

The system can be initialized with well defined descriptions for common services. Then the users are given the choice of creating their own format of the service description, or of using previously defined format of the same service if it exists. They may create a new service attribute
or use an existing one. In addition, they may choose the type of the attribute (either a multi-choice single selection mode attribute, multi-choice multiple selection mode attribute or unlimited values attribute) with the ability to customize the existing attributes where the effect appears only in their local service description. This means that even if an attribute exists in the general template and is used by other users, the changes made by the current user only affect the attribute in the current user service description (local template). The modified attribute, however, is made available for use by other users (if they desire to do so). This design philosophy is implemented by maintaining a flexible storage structure, where attributes and their corresponding values are stored as separate entities. Each service description references specific attributes and values; so that the addition of new values to the attributes or the creation of new attributes doesn’t appear in a service description unless the latter references it. This structure indicates that the system keeps track of the usage count for each attribute or value and the service which uses it.

The above specification raises the following problems:

On the demander side (i.e. the person who determines the required service description) the following questions should be answered:
1. How to deal with different service attributes which are related to the same concept?
2. How to handle badly defined attributes?

On the supplier side (i.e. the person who responds to a demand) the following questions should be dealt with:
1. How to create the service description and what if there are many attributes to fill?
2. New service descriptions may be created by demanders after the supplier creates his service description.

Solutions to problems number 1 and 2 on the demander side:

The user does not use an attribute or a value unless it is useful for her/him; and since the system has a count for each attribute and value, this could be used as an indication of the suitability of the attribute and consequently be recommended by the system. When the user desires to use an existing attribute with a service type that has been specified by other demanders, the most used attributes are displayed first. The user is free to choose these recommended attributes. In the case of multi-choice single mode or multi-mode attributes all of their values are displayed so that the user is capable of selecting a subset of the values of a given attribute.

Solutions to problem 1 on the supplier side:

The supplier may create his service description by entering the values of the attributes created by a demander looking for suppliers with the same service type as her/his. The supplier may define a general service description by entering values for all the used attributes in the service description, or by using a subset of the attributes, or even by using all of the attributes in a specific service description for a single demander. The frequently used attributes are displayed first for the supplier.

Solutions to problem 2 on the supplier side:

The supplier may instantiate a software agent that tracks the new attributes related to a specific service type. This agent would then prompt the user to enter values for these attributes after the supplier logs in or sends them to her/his email.
2.4 Reusability

When the demander defines a new attribute for the required service description, it’s stored in a database, so that it could be used by other demanders. Even the values related to multi-choice attributes could be reused. The system maintains statistics about the attributes (in particular their usage counts) which indicate the most frequently used attributes or the most used choices or values of a given attribute. This has the following advantages:

1. Guide the demander in the process of service description creation: if the system contains previously defined service descriptions then the system would suggest the most frequently used attributes or values of attributes when the demander desires to reuse existing ones.
2. Alert the supplier about the importance of most frequently used attributes or values of attributes so that the user may in the future try to provide these attributes or values in his service.

For example, consider a job matchmaking system, where an employer (demander) is looking for a programmer with the appropriate characteristics (attributes), s/he finds that the attribute “Programming Languages” is less used than the attribute “Personal Characteristics” where they are represented as a collection of checkboxes – to indicate the multi-mode selection nature- so s/he gets an indication that maybe the personal characteristics such as creativity are more important than the number of programming languages the programmer knows. In addition to that, when a programmer wants to create his resume, s/he realizes according the usage counts that the most required programming language (attribute value) in the domain is C++ (for example) so s/he decides to learn more about this language to improve her/his chance of getting a job.

Another example is a system such as eBay where a user tries to find a supplier for a certain product with specific characteristics. Suppose that a demander is searching for a desktop computer; to create the description of the product, s/he decides to use existing attributes. The system indicates that the most two used attributes are a dropdown list and a checkbox, where they are labeled respectively by “Warranty Years” and “Does this product follow the ISO standard”, with "0", "1", "2" and "3" as values of the Warranty Years dropdown list attribute. This could give the user an indication about the importance of the two previous attributes. On the other hand, after the supplier recognizes the high usage counts of the two attributes, s/he may try to increase the number of warranty years and ensure that her/his products follow the ISO standard.

3. A Job Matching Prototype System

We have built a job matchmaking prototype system that applies our model of dynamic attributes, weight based ranking and reusability. The employers are able to define their vacancy descriptions represented by form fields either by defining new attributes or reusing existing ones. They are also able to choose between three different types of attributes, namely, multi-choice single selection mode (Dropdown List), multi-choice multiple selection mode (collection of Checkboxes we called it Checkboxes Group), and unlimited values attribute (TextBox), this is combined with the ability to define the weight for each value or item of the Dropdown List or the weights for each Checkbox of the Checkboxes Group. The employer is able to reuse and customize existing multi-choice attributes; this means that s/he is able to choose some of the values from the Dropdown List (i.e. the user is not forced to use all of the values of the Dropdown List). The same applies for the Checkboxes Group.
The employer is also able to view the seekers’ resumes ranked in descending order of their proximity to her/his job description. These resumes (or applications) are created by filling at least one attribute of the employer job description. Also the system is capable of providing an explanation of how the ranking was derived. It also allows the employer to see each individual seeker’s resume along with the weights s/he gets for each attribute.

The system displays the usage count for each attribute and even values of that attribute. This makes it easy for the employer to see which attribute is used most. In our example, which programming language was most used in the job description, besides, the system displays the average weight of each value.

4. System Architecture

The architecture of the prototype is three-tier architecture as shown in Figure 1:

- **Database Tier**: this tier stores the data; it also contains information about the attributes of the services' descriptions.
- Logic Tier: this tier processes information and acts as an intermediate tier between the other two tiers, it also retrieves the information from the database and calculates their ranks.
- Presentation Tier: this tier displays the data and the results in a suitable format.

![Figure 1: The Prototype Architecture](image)

4.1 Tools and Programming Languages

The prototype was developed using Asp.net 2.0 with Visual Studio .Net 2005 and SQL Server 2005 for creating and storing the database. All the functions of the systems are governed by an easy to use GUI.

4.2 Database Tables

The system maintains the following tables:
- Employer and Seeker Tables (Demander and Supplier): these contain personal information about the job seeker and the company. For example the Employer table may contain the following information: Company Name, Main Business Area, Country, City, and Contact Information.
4.3 Data Flow in the System

Figure 2 shows the data flow in the system from the demander (employer) and from the job seeker (supplier) point of views. The employer starts by registering with the system. This process is performed once by every employer wishing to use the system. On subsequent uses of the system, the employer only logs onto the system and creates a vacancy description. This description is stored in the database. Also, during this process the employer may benefit from previously created vacancy announcements. In addition to that, the employer is capable of defining the attribute weights and the ranking strategy.

The job seeker also registers with the system for one time. On subsequent uses the seeker just logs onto the system; the seeker may create a resume which is consequently stored in the database. S/he may browse through stored vacancies or resumes to obtain a feeling of what constitutes a good resume. When seekers post their resumes as a response to a certain vacancy announcement, the system generates the set of matches for that vacancy sorted in decreasing order of their closeness to the demander request. This result is returned to the employer who notifies these matches for further actions.

5. Conclusions

This paper has presented a design philosophy and a prototype system for matchmaking. The design philosophy is based on dynamic attributes, a flexible scheme for determining weights of attributes and reusability. Dynamic attributes indicate that the service/demand supplier can define the attributes that s/he sees appropriate for the task at hands. This means that the provider is not confined to a set of pre-determined attributes. Weights of attributes are defined in the local template of the service/demand provider meaning that identical attributes that appear in the description of different services need not have the same weight. Also, reusability was a major goal in the proposed model where users may benefit from previously defined service descriptions. The above ideas were tested on a job matchmaking domain where a prototype was created to test the viability of the proposed ideas. It performed very well and we believe that the proposed ideas have a potential in the matchmaking field.
Figure 2: Data Flow in the System

References


