Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.5, 2012

Efficient Retrieval of Web Services Using Prioritization and Clustering

A.Aroj Prasad* B.Meena B.Udaya Kumar V.Kartheek
Department of Information Technology, ANITS, Andhra Pradesh, India.
*E-mail: arojprasad@gmail.com

Abstract
WEB services are software entities that have a well defined interface and perform a specific task. Typical examples include services returning information to the user, such as news or weather forecast services. A web service is formally described in a standardized language (WSDL). The service description may include the parameters associated with web services like input , output and quality of service. As web services and service providers proliferate, there will be a large number of candidate, and likely competing, services for fulfilling a desired task. Hence, effective service discovery mechanisms are required for identifying and retrieving the most appropriate services. The main contributions of our paper are summarized as follows; we propose and implement a method for determining dominance relationships among service advertisements that simultaneously takes into consideration multiple PDM criteria. We introduce a method for prioritization and clustering web services based on similarity measures using efficient algorithms
Keywords : Web Service , PDM , dominance score ,TKDD, clustering .

1.Introduction

Existing service matchmakers typically determine the relevance between a web service advertisement and a service request by computing an overall score that aggregates individual matching scores among the various parameters in their descriptions. Two main drawbacks characterize such approaches. First, there is no single matching criterion that is optimal for determining the similarity between parameters. Instead, there are numerous approaches ranging from Information Retrieval similarity measures up to semantic logic-based inference rules. Second, the reduction of individual scores to an overall similarity leads to significant information loss.

Assume the existence of a repository that contains a large number of advertised service descriptions. In a typical scenario, a user provides a complete definition of the requested service, and issues a discovery query. The repository, then, employs a matchmaking algorithm to identify services relevant to the user's request. Note that perfect matches, i.e., services with the same description as the request, are seldom found. Furthermore, even when a perfect match exists, it may not constitute the most desirable option, e.g., the service is currently unavailable. For these reasons, given a request, the matchmaking algorithm needs to consider a potentially large number of partial matches, and to select the best candidates among them.

The basic idea behind our implementation is to define features that are constrained to enhance the capabilities of existing web search engines for efficient web services retrieval using multiple criteria dominance relationship which is limited to provide the following features

- Retrieving Web service description file of web services
- Capture the input and output parameters of WSDL File.
- Compare input and output parameters of WSDL File with the parameters provided through the user interface using the similarity measures namely Cosine and Jaccard similarity
- Compute Dominated Score depending on similarity measures
- Find service priorities using TKDD Algorithm.
- Cluster the retrieved web services.

2.Related work

As we move from a Web to Web services, enhancing the capabilities of the current Web search engines with effective and efficient techniques for Web services retrieval and selection becomes an important issue. Existing service matchmakers typically determine the relevance between a web service advertisement and a service request by computing an overall score that aggregates individual matching scores among the various parameters in their descriptions

2.1 Web Service

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed

using HTTP with an XML serialization in conjunction with other Web-related standards. The purpose of a Web service is to provide some functionality on behalf of its owner a person or organization, such as a business or an individual. The provider entity is the person or organization that provides an appropriate agent to implement a particular service. The basic Web services platform is XML + HTTP for web service components . All the standard Web Services works using following components

- WSDL (Web Services Description Language.)
- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)

2.1.1 Web Service Description Language(WSDL)

WSDL stands for Web Services Description Language. WSDL is an XML based protocol for information exchange in decentralized and distributed environments. WSDL is the standard format for describing a web service. WSDL definition describes how to access a web service and what operations it will perform. WSDL is a language for describing how to interface with XML-based services. WSDL is an integral part of UDDI, an XML-based worldwide business registry. WSDL is the language that UDDI uses. WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.DDI uses. WSDL breaks down Web services into three specific, identifiable elements that can be combined or reused once defined. Three major elements of WSDL that can be defined separately and they are:

- Types
- Operations
- Binding

A WSDL document has various elements, but they are contained within these three main elements, which can be developed as separate documents and then they can be combined or reused to form complete WSDL files. Following are the elements of WSDL document. Within these elements are further sub elements, or parts:

Definition: It defines the name of the web service, and must be the root element of all WSDL documents.

Data types: The data types - in the form of XML schemas or possibly some other mechanism - to be used in the messages.

Message: An abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation. It consist of parameters and their types .

Operation: The abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message.

Port type : An abstract set of operations mapped to one or more end points, defining the collection of operations for a binding;

Binding: The concrete protocol and data formats for the operations and messages defined for a particular port type.

Port: A combination of a binding and a network address, providing the target address of the service communication.

Service: A collection of related end points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.

The main structure of a WSDL document looks like this:

```
<definitions>
<types>          definition of types.....          </types>
<message>      definition of a message...          .</message>
<portType>          <operation>
definition of a operation.....
</operation>
</portType>
<binding>      definition of a binding.... </binding>
<service>      definition of a service.... </service>
</definitions>
```

2.1.2 Universal Description Discovery Integration (UDDI)

UDDI stands for Universal Description, Discovery and Integration. UDDI is an XML-based standard for describing, publishing, and finding Web services. UDDI is a specification for a distributed registry of Web services. UDDI is platform independent, open framework. UDDI can communicate via simple object access protocol (SOAP) , common object request brokers architecture (CORBA), Java remote method invocation (RMI) Protocol. UDDI uses WSDL to describe interfaces to web services. UDDI is seen with SOAP and WSDL as one of

the three foundation standards of web services. UDDI is an open industry initiative enabling businesses to discover each other and define how they interact over the Internet.

2.1.3 Simple Object Access Protocol(SOAP)

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.

2.2 Web Service Discovery:

Consider the following typical Web service discovery scenario. The user provides a complete definition of the desired service and poses a query on a system maintaining a repository of advertised service descriptions. Alternatively, the user could specify a desirable service, e.g., among previous results, and request similar services. Then, the search engine employs a matchmaking algorithm identifying advertisements relevant to the user's request. A lot of recent work has focused on defining objectively good similarity measures capturing the degree of match between a requested and an advertised service. Typically, this process involves two steps: (i) selecting a criterion for assessing the similarity of service parameters, and (ii) aggregating individual parameter scores to obtain the overall degree of match between a request and an advertisement.

The first step involves the estimation of the degree of match between parameters of the request and the advertisement. There are two paradigms for assessing the match among parameters. The first, treats parameter descriptions as documents and employs basic Information Retrieval techniques to extract keywords. Subsequently, a string similarity measure is used to compute the degree of match. The second paradigm follows the Semantic Web vision. Services are enriched by annotating their parameters with semantic concepts taken from domain ontologies. Then, estimating the degree of parameter match reduces to a problem of logic inference: a reasoner is employed to check for equivalence or subsumption relationships between concepts. Both paradigms share their weaknesses and strengths. Regarding the former techniques, keyword-based matchmaking fails to properly identify and extract semantics since service descriptions are essentially very short documents with few terms. On the other hand, the latter techniques face common Semantic Web obstacles, e.g., the lack of available ontologies, the difficulty in achieving consensus among a large number of involved parties, and the considerable overhead in developing, maintaining an ontology and semantically annotating the available data and services. More recently, hybrid techniques for estimating the degree of parameter match have appeared, taking into account both paradigms. Still, the common issue with all approaches is that there is no single matching criterion that optimally determines the similarity between parameters. In fact, different similarity measures may be more suitable, depending on the particular domain or the particular pair of request and offer. Therefore, we advocate an approach that simultaneously employs multiple matching criteria. The second step in matchmaking deals with the computation of the overall degree of match for a pair of requested and advertised services taking into consideration the individual scores of corresponding parameters.

2.3 Parameter Degree Match(PDM)

Both prioritization and clustering require as a first step the matchmaking algorithm to assign to each considered parameter a parameter degree of match (PDM) with respect to the requested service. Then, a service degree of match (SDM) can be computed as an aggregate of the individual PDMs. Various approaches for combining PDMs exist. Consequently, service degree of match based prioritization may assign very low ranks, for example, to services with a single bad matching parameter. Similarly, SDM based clustering fails to construct representative groups. PDMs provide a finer granularity for the discovery and selection process.

2.4 Service Dominance Score

2.4.1 Dominated Score

Given an instance u, we define the dominated score of u, denoted by dds as

$$u.dds = \sum_{V \neq U} \frac{|\{v \in V \mid v \succ u\}|}{|V|}$$

Hence, u.dds considers the instances that dominate u. The dominated score of an object U is defined as the (possibly weighted) average of the dominated scores of its instances:

$$U.dds = \sum_{u \in U} \frac{u.dds}{|U|}$$

The dominated score of an object indicates the average number of objects that dominate it. Hence, a lower dominated score indicates a better match.

2.4.2 Dominating Score

Given an instance u, we define the dominating score of u, denoted by dgs, as:

$$u.dgs = \sum_{V \neq U} \frac{|\{v \in V \mid u \succ v\}|}{|V|}$$

Thus, u.dgs considers the instances that u dominates. The dominating score of an object U is defined as the (possibly weighted) average of the dominating scores of its instances:

$$U.dgs = \sum_{u \in U} \frac{u.dgs}{|U|}$$

The dominating score of an object indicates the average number of objects that it dominates. Hence, a higher dominating score indicates a better match.

2.5 Prioritization using TKDD

Prioritization entails assigning a score to each advertisement, quantifying its suitability for the given request. Given that users typically view only a few top search results, it is important that useful results appear high in the list. Similarly, in fully automated scenarios, where a software agent automatically selects and composes services to achieve a specific task, only the few top ranked results are typically considered. The problem of prioritization web services entails computing the scores of services and returning the top-k highest ones. Prioritization of services is done based on the dominated and dominating scores. Transaction Knowledge Data Discovery (TKDD) Algorithm is used for prioritization of web services.

2.6 Clustering

Clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters. Clustering organizes advertisements so that services within a cluster provide similar matches with respect to the request. Since several parameters are involved in the matchmaking process, finding a service that provides a high degree of match for all parameters is difficult; instead, it is often needed to decide between different trade-offs. Clustering the search results allows the user to identify an interesting advertisement and then browse similar results, i.e., those found within the same cluster. The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression. Cluster analysis has been widely used in numerous applications, including market research, pattern recognition, data analysis, and image processing.

3 . Implementation

3.1 Retrieving the WSDL File:

Each Web Services has a description file associated with it written using Web Service Description Language. This file contains the input output parameter associated with Web Service, how a web service needs to invoked ,what are the operations a web services can perform and also some Quality of Services parameters. Retrieving of this file is done using URL connection class after which all input and output parameters of WSDL file collected.

3.2 Computing Cosine Similarity

Cosine similarity is a measure of similarity between two vectors by measuring the cosine of the angle between them. We convert the two strings to be compared into binary vector and thus apply cosine function to the vectors. The value lies between 0 and 1.Higher is the value better is the match. The value 1 corresponds to exact match between two string and the value 0 corresponds to totally different strings. The cosine of two vectors can be easily derived by using the Euclidean dot product formula: $a.b=|a|\ |b|\ \cos\Theta$

Given two vectors of attributes, A and B, the cosine similarity, θ, is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Computing Jaccard Coefficient

The Jaccard index, also known as the Jaccard similarity coefficient is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets: Given two objects, *A* and *B*, each with *n* binary attributes, the Jaccard coefficient is a useful measure of the overlap that A and B share with their attributes. Each attribute of A and B can either be 0 or 1. The total number of each combination of attributes for both A and B are specified as follows:

M11 represents the total number of attributes where A and B both have a value of 1.
M01 represents the total number of attributes where the attribute of A is 0 and the attribute of B is 1.
M10 represents the total number of attributes where the attribute of A is 1 and the attribute of B is 0.
M00 represents the total number of attributes where A and B both have a value of 0.
Each attribute must fall into one of these four categories, meaning that M11 + M01 + M10 + M00 = n
The Jaccard similarity coefficient, J, is given as M11 / (M01 + M10 + M00)

3.3 *Prioritization By Dominated Score:*

This module computes top-k web services according to the dominated score criterion. The goal is to quickly find, for each object, other objects dominating it, avoiding an exhaustive comparison of each instance to all other instances. The algorithm maintains three lists, Imin, Imax, and I, containing, the minimum bounding instances, the maximum bounding instances, and the actual instances of the objects. The instances inside these lists are sorted and are examined in descending order. The results are maintained in a list R sorted in ascending order of dds. The algorithm uses two variables, ddsMax and minValue, which correspond to an upper bound for dds, and to the minimum value of the current kth object, respectively. for an object U, we are interested in objects that dominate it, we search only for instances that are prior to those of U in I. Since, the top matches are expected to appear in the beginning of I, this significantly reduces the search space.

3.4.1.TKDD Algorithm

The first algorithm, hereafter referred to as TKDD, computes top-k Web services according to the dominated score criterion, dds. The goal is to quickly find, for each object, other objects dominating it, avoiding an exhaustive comparison of each instance to all other instances. The algorithm maintains three list, Imin, Imax, and I, containing, respectively, the minimum bounding instances, the maximum bounding instances, and the actual instances of the objects. The instances inside these lists are sorted by $F(u) = \Sigma_i u[i]$ and are examined in descending order. The results are maintained in a list R sorted in ascending order of dds.

The algorithm uses two variables ddsMax and minValue, which correspond to an upper bound for dds, and to the minimum value of the current k-th object, respectively. Given that, for an object U, we are interested in objects that dominate it, we search only for instances that are prior to those of U in I. Since, the top matches are expected to appear in the beginning of I, this significantly reduces the search space. The basic idea is to use the bounding boxes of the objects to avoid as many dominance checks between individual instances as possible. After k results have been acquired, we use the score of the k-th object as a maximum threshold. Objects whose score exceeds the threshold are pruned. In addition, if at some point, it is guaranteed that the score of all the remaining objects exceeds the threshold, the search terminates. More specifically, the algorithm, proceeds in the following six steps.

Step 1. Initializations The result set R and the variables dds Max and min value are initialized. The lists I min, Imax, and I are initialized, and sorted by F(u). Then the algorithm iterates over the objects, according to their maximum bounding instance.

Step 2. Termination condition. If the F() value of the current umax does not exceed the minimum value of the current k- th object, the result set R is returned and the algorithm terminates.

Step 3. Dominance check object-to-object. For the current object U, the algorithm first searches for objects that fully dominate it. For example, in the case of the data set of Figure 1, with a single dominance check between bmax and amin, we can conclude that all instances b1, b2 and b3 are dominated by a1, a2 and a3. According to, only objects with F(vmin) >F(umax) need to be checked. If a vmin is found to dominate umax, then the score of U is increased by 1, and the sum of the new score and the score of V is compared to the current threshold,

ddsMax. If it exceeds the threshold, the object is pruned and the iteration continues with the next object. In this case, the score of the object is propagated to its instances for later use. Otherwise, the score of the object is reset, to avoid duplicates, and the search continues in the next step.

Step 4. Dominance check object-to-instance. This step searches for individual instances v that dominate U. For example, in Figure 1, a dominance check between dmax (which coincides with d1) and c1 shows that all instances d1, d2, and d3 are dominated by c1. As before, only instances with F(v) > F(umax) are considered. If an instance v is found to dominate umax, then the score of U is increased by 1/M, where M is the number of instances per object, and the sum of the new score and that of v is compared to the current threshold, ddsMax.

Step 5. Dominance check instance-to-object . If the object U has not been pruned in the previous two steps, its individual instances are considered. Each instance u is compared to instances vmin, with F(vmin) > F(u). If it is dominated, the score of u is again increased by 1/M, and the threshold is checked. In Figure 1, this is the case with d3 and bmin.

Step 6. Dominance check instance-to-instance. If all previous steps failed to prune the object, a comparison between individual instances takes place where each successful dominance check contributes to the object's score by 1/M2

Step 7. Result set update. If U has not been pruned in any of the previous steps, it is inserted in the result set R. If k results exist, the last is removed. After inserting the new object, if the size of R is k, the thresholds ddsMax and min Value are set accordingly.

3.4 *Clustering Web Services using K-Means Algorithm:*

K-Means Clustering is used for clustering the web services where the initial centroids are chosen randomly and points near to the initial centroids are formed as one cluster. After one iteration means of the points in one cluster are computed and new centroids are formed. Again the points closer to the new centroids are formed as one cluster. This process is repeated until the centroid values does not change for two consecutive iterations where similar web services are clustered into one cluster.

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function,

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

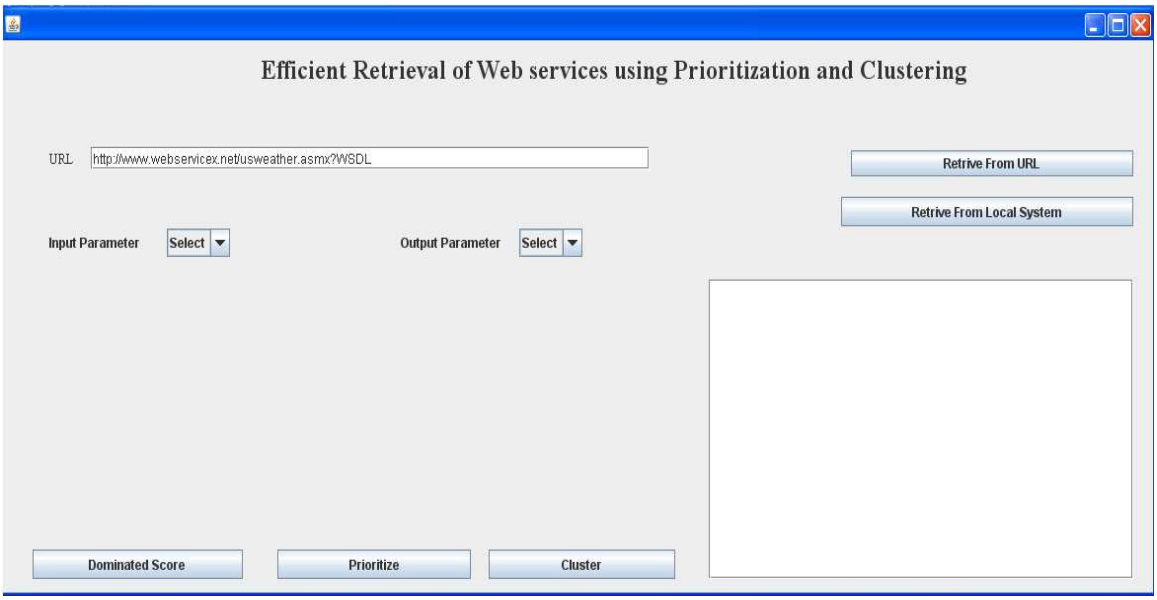
where $|X_i^{(j)} - c_j|^2$ is a chosen distance measure between a data point $X_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the n data points from their respective cluster centers.

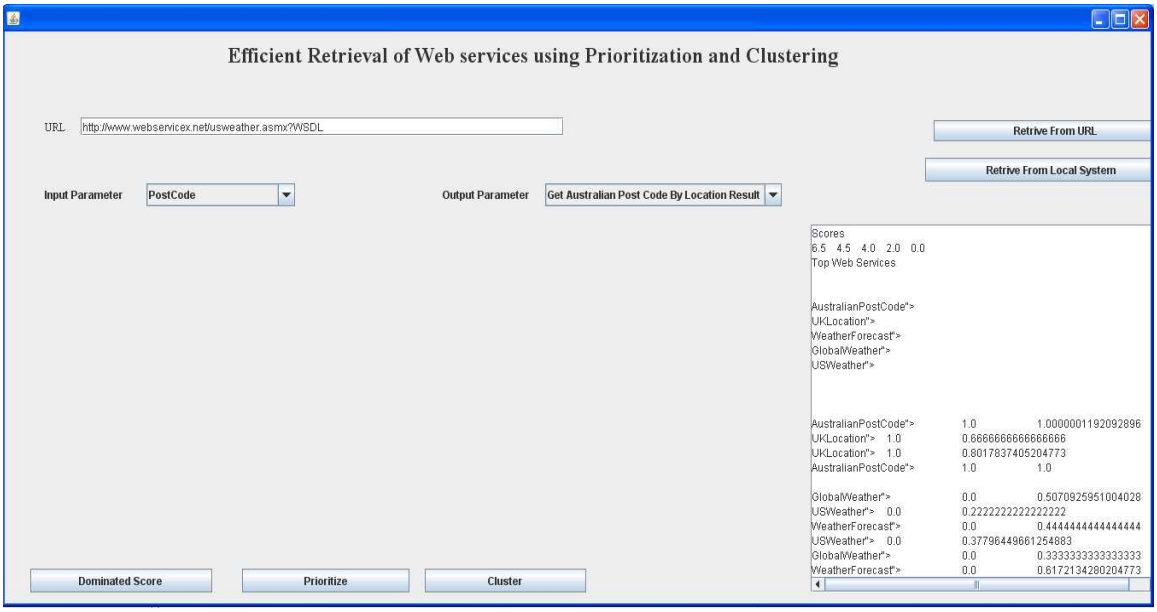
The algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

The implementation of our system is done in Netbeans 7.0 using JAVA . Netbeans is a tool that was originally designed to simplify the coding involved in implementation. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java is currently one of the most popular programming languages in use, particularly for client-server web applications.

4.Screen Shots





5. Conclusion

In this Paper, prioritization and clustering of web service is based on the notion of dominance, which apply multiple similarity measures aggregating the match scores of individual service parameters. TKDD algorithm has been used for this purpose which retrieves the top K-Dominant services. K-means algorithm is used for selecting the most representative services for clustering, so that the produced clusters reflect the trade-offs between the matched parameters.

6. References

[A] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic Skylines on Uncertain Data," Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB), pp. 15-26, 2007.
[B] D. Skoutas, A. Simitsis, and T. Sellis, "A Ranking Mechanism for Semantic Web Service Discovery," Proc. IEEE Services Computing Workshops (SCW), pp. 41-48, 2007.
[C] U. Bellur and R. Kulkarni, "Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching," Proc. IEEE Int'l Conf. Web Services (ICWS), pp. 86-93, 2007.
[D] J. Ma, Y. Zhang, and J. He, "Efficiently Finding Web Services Using a Clustering Semantic Approach," Proc. Int'l Workshop Context Enabled Source and Service Selection, Integration and Adaptation (CSSSIA), p. 5, 2008.
[E] I. Bartolini, P. Ciaccia, and M. Patella, "Efficient Sort-Based Skyline Evaluation," ACM Trans. Database Systems, vol. 33, no. 4, pp. 1-45, 2008.
[F] K.C.K. Lee, B. Zheng, H. Li, and W.-C. Lee, "Approaching the Skyline in Z Order," Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB), pp. 279-290, 2007.
[G] M.L. Yiu and N. Mamoulis, "Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data," Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB), pp. 483-494, 2007.
[H] W.-T. Balke, U. Gu¨ ntzer, and C. Lofi, "Eliciting Matters— Controlling Skyline Sizes by Incremental Integration of User Preferences," Proc. 12th Int'l Conf. Database Systems for Advanced Applications (DASFAA), pp. 551-562, 2007.
[I] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting Stars: The k Most Representative Skyline Operator," Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE), pp. 86-95, 2007.

Example Service with Multi Criteria parameter degree of match

Service	Parameter	Measure1	Measure2	Measure3
A	Input	0.86	0.99	0.93
	Output	0.82	0.96	1.00
B	Input	0.70	0.65	0.65
	Output	0.70	0.86	0.71
C	Input	0.85	0.88	0.75
	Output	0.85	0.84	0.60
D	Input	0.76	0.67	0.56
	output	0.76	0.64	0.62

Table 1: Parameter degree of match

The table illustrates the complete scenario. It consists of 4 web services A,B,C,D each having input and output parameters. Three measures are used like Cosine Similarity, Jaccard Similarity and Jensen-Shannon information divergence based similarity. The values shows the match scores for each web services