International Journal of Multidisciplinary and Current Research

Research Article

ISSN: 2321-3124
Available at: http://ijmcr.com

Web Service Integration using Knowledge Discovery in Services (KDS)

Mr. Praveenkumar Arjun Patel[†] and Prof. Umesh Laxman Kulkarni[‡]

[†]PG Student, Dept. of CSE, DYPCET, Shivaji University, Kolhapur, India
[‡]Department of Computer Engineering, Mumbai University, Mumbai, India

Accepted 15 June 2015, Available online 21 June 2015, **Vol.3 (May/June 2015 issue)**

Abstract

Web Service Integration now receives a great deal of interest from academia and industry. Service Integration is the work of merging the resulting data of various complementary web services into a general situation. This approach is undertaking in relation to the discovery of new types of information. Prior to service integration procedures can be executed, it is needed to expect which web services are required candidates. Knowledge Discovery in Services (KDS) process is dedicated to address a repository of open services that do not contain semantic annotations. In these conditions, focused methods are needed to find out equivalences among open services with practical exactness. Web services data are clustered by using KNN algorithm. Categories have created as per similarity scores. Filtering receives data from either Clustering or Categorization OR from both. Finally Composite Service Output put on views multiple predictions for particular user. It starts a bottom-up process for KDS that adapts to the environment of services for which it operates.

Keywords: *Categorization, Clustering, Composite Service Output, Filtering, Service Specification & Equivalence Processing, Web services*

1. Introduction

Plenty of modular software capabilities discoverable and usable by disparate organization are supported by Service-Oriented Computing (SOC). Underlying SOC has features, such as the Web Service Description Language (WSDL), and the invocation techniques maintained by SOAP or REST that translate machine interfaces and communications. Such open approaches present situation where web services are used.

Prototype of service-oriented computing is Web 2.0 that wraps the idea of it. Arrangement of fundamental model and Web 2.0 is gains for anyone user to be the major stakeholder. As a result of association technologies and market oriented environments user will interact perfectly such as, any user may work web services in a most appropriate style to their daily activities. The integration of the resultant data into a common vision can be of important while business process composition is unnecessary.

Service integration is the analogous execution of two or more services to create an integrated data providing with a more complete description about some object or task.

2. Literature Survey

Now a day's service integration receives a great deal of knowledge from academia and industry. Tools and techniques that instrument the service integration process and subsequently conceive of the results are involved in much of the current work. This model of work typically narrates to the great research area of data and information combination. Knowledge Discovery in Databases (KDD) [1] is initial idea from which we considers knowledge discovery but instead of databases or data mining, we believe new knowledge that can be attained when aggregating complementary services.

Knowledge Discovery in Services (KDS) [1] is a systematic process for discovering web service candidates for service integration that may ultimately discover resourceful knowledge. In this approach, there is an adapted development life cycle that software engineers can use to generate new applications based on service integration methods. KDS also find out the characteristics of the web service specifications that are most precious for determining service integration qualification.

In this paper the process that intersects the areas of data integration and service mashup. This representation for discovering data from web service provisions is similar to the well-established approaches associated with KDD. This paper discusses the likenesses of the several processes. As well, the modern approaches related to the underlying KDS areas of equivalence processing, clustering, categorization and filtering [1].

Although the area of data integration has had an ancient background, the utilization of these methods to

accomplish mashups has only just newly started to be addressed. Most of the work in this area deals with the tools and environments. That preserves the visualization and presentation of mashup results [2], [3]. Other projects depict enabling techniques for preparing services for mashup [4]. There are also projects that investigate the policy for protecting data in mashup environments and instituting enterprise policy for modernizing systems using the information resulting from successful mashups [5].

3. Research Method

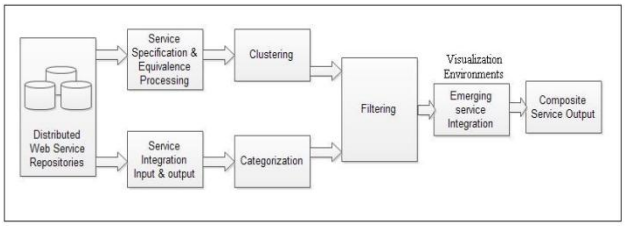


Figure 1: Proposed Web Service Integration Architecture

4. Equivalence Processing

In Conventional software engineering places the match of software interface for has explored. Our approach compares the similarity search of earlier approach by making use of markup language interface which is available within service oriented calculating domain. Also the advances to service integration are similar to the fundamental techniques for the discovery and composition of web services. Semantic and syntactic methods are the two common approaches to composition, utilization and perhaps service integration. Usually semantic advances maintain the integration of web services and functionality of semantic description.

Algorithm similarly attempts to understand the trends of the software developers such as Subsumption Relation, Common Subsets and Abbreviations in Naming that name their web services. In order to understand these trends, we manually download and verified the functionality of many open web services. Web services arrive from a number of web services repositories.

User interfaces are main query and sub queries. The intended work searches in distributed web service repositories of Universal Description Discovery and Integration (UDDI) as per the query. This can be a service name, operation name, type name and expressive fields from the service specification. Tendency-based Syntactic Matching-Levenshtein Distance and Letter-Pairing (TSM-LP) [1] algorithm is used for equivalent processing between input query and specific part of web service.

Table 1: TSM-LP Pseudo code

Tendency-based Syntactic Matching- Levenshtein Distance/Letter Pairing Algorithm (TSM-LP)
TSM-LP(Si , Sj) TSM-L function
LD (Si , Sj) Levenshtein distance function
FT1(Si) Trend-based threshold
FT2(Si) Trend-based threshold for letter pairing
Si, SjTwo strings for comparison
Length() String length functions
CS Web services category (such as Business)
FT1(Si) *temp = [(Length(Si) * 2) / 3] – 2* *return temp* *FT2(Si)* *temp = Sensitivity (CS)* *return temp*
TSM-LP(Si , Sj) *if (LD(Si, Sj) <= FT1(Si)) or* *(LP(Si, Sj) >= FT2(Si)) or* *(Si ⊆Sjor Sj⊆Si)* *(Si > 3 and Sj> 3) and* *(Si < 15 and Sj< 15)* *return TRUE* *else* *return FALSE*

4.1 Discovering styles in Web Service Message Naming

KDSapproach uses specialized parsing tools to assess services from the bottom-up. We required and created styles across the release web services. The three most general styles are listed as follows:

4.1.1. Subsumption Relationships: The strong style for web service developers is to use part names based on common names. Analogous messages plan to have well-built subsumption relationships when using common names. For example, there are equivalent services where name = fname; name = middle name, and name = user name.

4.1.2. Common Subsets: This is similar to subsumption relationships, various piece of web service names are relayed to have common subsets. In support of occurrences, we found equal element names last name = user name.

4.1.3. Abbreviations in Naming: Another strong trend was common names reduced into abbreviations. For example, circuit = ckt or communication = comm.

4.2. Using Natural Language Approaches to Exploit Styles in Message Matching

Several syntactic advances were related to take advantage of Style 2 and 3 when matching services**.**

The Levenshtein distance (LD) (besides called the edit distance) assesses of similarity between two strings. LD is the smallest number of deletions, insertions, or substitutions which are needed to transform an original

string into a target string. The bigger the LD, the more difference of the strings depends on LD. From preexisting supplies, we adapt implementations of the LD algorithm from preexisting sources. The LD algorithm is doing well when assessing abbreviations with complete strings. LD is too successful for analogous strings that are changed to create uniqueness. There were a number of occurrences where zeros are substituted for the letter O or numbers are added to the end of a message name.

LD is helpful for analogous strings, but it is not useful for strings that have analogous subsets that do not have exact subsumption relations as in Style 1. For instance, last name and surname are the same but neither is a subset of the other. We employed the use of Letter Pairing for the account of instances of this nature. The Letter Pairing (LP) approach is an algorithm that can be used to match strings that have common subsets. Using the LP algorithm, two strings are separated into letter pairs. STR1 would be break up into ST, TR, and R1, and STR2 would be break up into ST, TR, and R2. A ratio of the number of identical pairs and the total number of pairs is calculated. This percentage is exercised to assess the similarity of the two strings.

4.3. Different Approaches for Syntactical Message Matching

The overarching matching algorithm is called Tendency-Based Syntactic Matching- Levenshtein Distance and Letter Pairing (TSM-LP). This algorithm exploits the previously mentioned trends using LD and LP. However, as a part of the service integration experimentation we will assess using the following four methods of syntactical matching:

- Equality: This process checks if two parts of message are syntactically the equal.
- Subsumption: This process checks if the two parts of message have a subsumption relation (i.e., message part 1 is a subset of message part 2 or vice versa). Note: Subsumption includes equality.
- Levenshtein Distance (TSM-L): This process adds to equality and subsumption by also testing to see if message parts are similar based on edit distance. Three thresholds based on changeable strictness manage the algorithm. The three thresholds are determined based on the degree of self-similarity for the particular category for which the service is related.
- Letter-Pairing (TSM-P): This process extends equality and subsumption by inspecting similarity using letter pairing. Strictness thresholds are based on repository similarity and align with TSM-L.
- Levenshtein Distance/Letter-Pairing (TSM-LP): This process merges all of the prior mentioned methods.

Figure 2: Process Query for Equivalence Processing

5. Clustering

5.1. Convert XML file to Database table

Once equivalences are identified between input query and specific parts of web services, web services are linked together by input and output messages. XML file is generated for data obtained from web services with convert XML file to Database table.

5.2. Create Clusters

The K-Nearest Neighbour algorithm is one of the foundation algorithms in instance based studying. This algorithm begins from relating every instance to its corresponding point in an n-dimensional space. The closest values of an instance are determined using the formula for Euclidean distance. Therefore, consider an example x and $a_r(x)$ the value of the attribute r of the example x, the subsequent vector that depicts this instance can be defined:

$$<a_1(x), a_2(x),\ldots,a_n(x)>$$

At this instant the distance between two instances x_i and x_j is defined like this:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

In the nearest neighbour finding, the goal function can present discreet values as well as real ones.

5.3. Cluster Creation

Cluster Rajarampuri

Create clusters by area wise in each city. Each area has allocated weight. Weight is increased by 1 as per consecutive areas in the particular city. Closer areas having smallest weight variation calculated using Euclidean Distance.Produce clusters using K-Nearest Neighbour Algorithm such as Rajarampuri cluster.

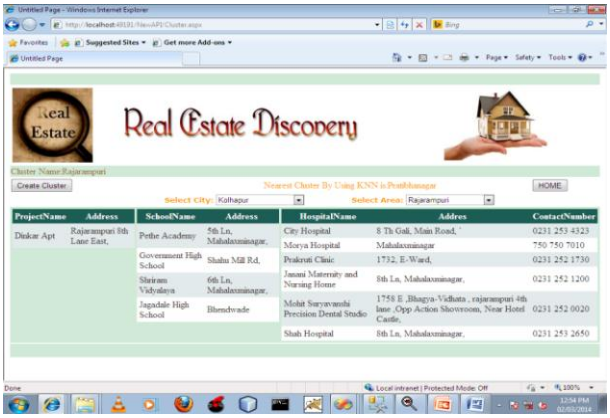


Figure 3: Cluster Rajarampuri

6. Categorization

Categorization creates categories using Categorization On Pairing (COP) approach. Each service is compared to every other service in the repository once. Compute approximately (N*(N-1)/2) unique similarity scores where N is number of Web Services. Unique similarity score sorted from largest to smallest. The pair of services with greatest similarity score makes a category for these two services.

6.1 Categorization on pairing Algorithm

COP algorithm has worked out as follows

- Algorithm starts at the pair of services with greatest similarity and makes category for these two services.
- Algorithm uses the second most similar pair of services and checks if either of the services has already been categorized.
- If one of them has been categorized the algorithm places the uncategorized service of the pair into the same category as the categorized service.(Line 7 & 17)
- If neither of the services has been categorized before, the algorithm new category for the pair and moves to the next most similar pair.(Line 26-34)

Figure 4 shows Categorization for Kothrud in Pune.

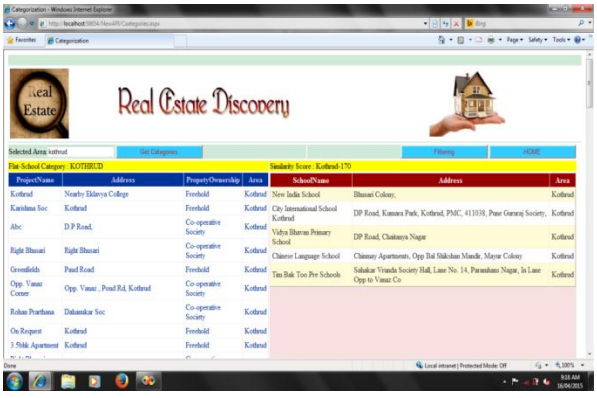


Figure 4: Categorization Kothrud

Table 2: Categorization on Pairing Algorithm

Pair = two pointers to services
Category = a collection of (similar) services
Pair [] **pairings** = sorted list of all unique service comparisons from most similar to least similar
Vector<Service> categorized = list categorized services
Vector<Category> cats = a vector of all the categories that have been created
cats.which(Service) = returns category containing a service

```
1:  private Vector<Category> buildFreeCategories()
2:    {
3:    for(Pair p : pairings)
4:      {
5:      Service1 = p.service1
6:      Service2 = p.service2
7:       If (categorized.contains(Service1)&&
8:                          !categorized.contains(Service2))
9:         {
10:
11:         Category hasService1 = cats.which(Service1);
12:         hasService1.addService(Service2);
13:         categorized.add(Service2);
14:         break;
15:        }
16:   //The second service has a category, the first does not
17:      else if (!categorized.contains(Service1)&&
18:                          categorized.contains(Service2))
19:         {
20:         Category hasService2 = cats.which(Service2);
21:         hasService2.addService(Service1);
22:         categorized.add(Service1);
23:         break;
24:        }
25:    //Neither service has a category
26:     else if (!categorized.contains(Service1) &&
27:                 !categorized.contains(Service2))
28:       {
29:      Category cat = new Category();
30:         cat.addService(Service1);
31:         cat.addService(Service2);
32:         categorized.add(Service1);
33 :        categorized.add(Service2);
34:         cats.add(cat);
35:        }
36:
37:   if(sorted.size() == servs)
38:   break;
    //all have been categorized, end the loop
    }
}
```

7. Filtering

The Filtering module within the KDS process attempts to predict which service integration predictions will add value for a particular user.

7.1. Multiple Predictions

Filtering is done by Prediction Algorithm. This approach takes data from either categorization or clustering or from both. The pseudo code for the Prediction algorithm is shown in Table 4.3.

Table 3: Prediction Algorithm

```
1.Prediction Algorithm:
2.P1 : FlatServiceData
3.P2 : SchoolServiceData
4.P3 : HospitalServiceData
5.intMaximumDataCount : Integer
6.P1,P2,P3 = Get Service Data From Cluster or Category

--Calculate Maximum Data
7.Interger intMaximumDataCount = 0;

8. IF P1.DataCount > 0
9.   intMaximumDataCount  = P1.DataCount
END IF
10. IF P2.DataCount > 0 && P2.DataCount > P1.DataCount
11. intMaximumDataCount = P2.DataCount
END IF
12.IF P3.DataCount > 0 && P3.DataCount > P2.DataCount
13. intMaximumDataCount = P3.DataCount
END IF
- -Get Filtered Data
14. IF intMaximumDataCount> 0
15. CompositeOutput : Data
16. Foreach Data (Area) in P1, P2, P3
17.       If P1:Data is present && P2.Data is present &&
P3.Data is present
18.       PredictionData P1P2P3 =
GetDataBySimilarityScore(P1.Data (Area),
19. P2.Data (Area), P3.Data (Area))
20.       ELSE If P1:Data is present && P2.Data is present
&& P3.Data is not present
21.             PredictionData P1P2 =
GetDataBySimilarityScore(P1.Data(Area),P2.Data (Area))
22.       ELSE If P1:Data is present && P2.Data is not
present && P3.Data is present
23.             PredictionData P1P3 =
GetDataBySimilarityScore (P1. Data (Area),P3.Data (Area))
24.       ELSE If P1:Data is not present && P2.Data is
present && P3.Data is present
25.             PredictionData P2P3 =
GetDataBySimilarityScore (P1. Data (Area),P3.Data (Area))
26.       END IF
27.       CompositeOutput : P1P2P3, P1P2, P1P3, P2P3
28. End Foreach
29   return CompositeOutput;
30. ELSE
31. Print : Data Not Found
32. END IF
```

Characteristics of Prediction algorithm is as follows

- Algorithm receives input from either clustering or categorization or from BOTH.
- Algorithm filters data area wise.
- Multiple predictions for particular user query.
- Prediction algorithm results add value to particular user.

Figure 5 shows filtered data of Kothrud area in Pune city.

Figure 5: Filtering Kothrud

8. Composite Service Output

8.1. Display Composite Service Output

The service integration for Composite Service Output within the KDS process has been given predictions will add value for a particular user.

8.2. Multiple Predictions given by Composite Service Output

Composite Service Output takes data from filtering. It uses visualization environment for predictions will add value for a particular user.

- Visualization Environment takes input from filtering.
- Service Integration is the act of combining the resulting data of various complementary web services into a common picture.
- Web Service Integration using Knowledge Discovery in Services an approach is promising with respect to the discovery of new types of knowledge.
- Multiple predictions displayed for particular user query.

Figure 6 shows results of web service integration called Composite Service Output for Kothrud area in Pune city.

Figure 6: Composite Service Output Kothrud

9. Results and Analysis

Fig.1 shows Proposed Web Service Integration Architecture. Table 1 contains pseudo code for Tendency-Based Syntactic Matching- Levenshtein Distance and Letter Pairing (TSM-LP) algorithm. Equivalence processing takes input query from user that query processes on web services such as Flat, Hospital and School. Fig.2 shows Equivalence Processing output for flats availability, in addition to close hospital and school for children in Rajarampuri at Kolhapur. Fig. 3 shows result of clustering using K-Nearest Neighbour Algorithm Fig 4 shows categorization by using Categorization On Pairing algorithm. Fig 5 Shows Filtering by using Prediction algorithm. Fig 6 shows Composite Service Output of service integration using Visualization Environment.

9.1. Composite Service Output result analysis:

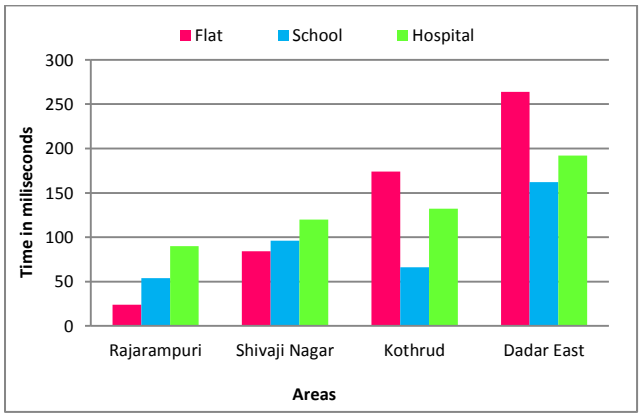
Table 4 shows web services such as Flat, School, Hospital v/s different areas in various cities requires time in milliseconds for composite service output.

Table 4: Composite Service Output result analysis

	Web Services		
Areas	**Flat**	**School**	**Hospital**
Rajarampuri	24	54	90
Shivaji Nagar	84	96	120
Kothrud	174	66	132
Dadar East	264	162	192

9.2. Composite Service Output Graph

Graph 1 shows Graphical representation of Areas on X - Axis v/s Time in milliseconds on Y-Axis for composite service output of web services.



Graph 1: Composite Service Output Graph

Conclusion and Future Work

Web Service Integration currently receives a great deal of attention from academia and industry. This body of work mostly narrates to the greater research area of data and information integration. Analogous to the primary notions of Knowledge Discovery in Databases (KDD), in proposed work also considers knowledge discovery but instead of databases or data mining, we consider new knowledge that can be attained when aggregating complementary services which we call Knowledge Discovery in Services (KDS)

The proposed system creates dynamically changing numbers of clusters by using KNN algorithm and creates categories by using Categorization on Pairing algorithm. Filtering data obtains from either clustering or categorization or from both by using Prediction algorithm. This system builds visualization environment for composite service output to discover new knowledge from complementary web services. Knowledge Discovery in Services (KDS) puts together proposed system very efficient use for web service integration.

Future work for this work to handle web services in multilingual approach. Another future work of Web Service Integration using Knowledge Discovery in Services (KDS) suggest that to create web services such as public transportation facilities, parks and restaurants close by flats. That will supportive for hunting flats in respective areas.

References

[1]. M. Brian Blake, Michael F. Nowlan,, (June 2011), Knowledge Discovery in. Services (KDS): Aggregating Software Services to Discover Enterprise Integrations, IEEE Transactions on, Knowledge and Data Engineering, vol.23, no.6, pp.889-901.

[2]. Praveenkumar Patel, UmeshKulkarni, (January 2015),Different Syntactic Methods and Clustering for Web Service Integration is published in International Journal of innovative Research in Computer and Communication Engineering, Volume 2, Issue 01.

[3]. S. Cetin, N.I. Altintas, H. Oguztuzun, A. Dogru, O. Tufekci, and S.Suloglu, (2007),A Integration-Based Strategy for Migration to Service-Oriented Computing, Proc. IEEE Int'l Conf. Pervasive Services

[4]. A. Jhingran, (2006), Enterprise Information Integrations: Integrating Information,Simply, Proc. 32nd Int'l Conf. Very Large Data Bases

[5]. M. Sabbouh, J. Higginson, S. Semy, and D. Gagne, (2007),Web IntegrationScripting Language, Proc. 16th Int'l Conf. World Wide We

[6]. J. Zou and C.J. Pavlovski, (Oct. 2007), Towards Accountable Enterprise Integration Services, IEEE Int'l Conf. E-Business Eng., pp 205-212.

[7]. M. Brian Blake, (Mar 2009), Knowledge Discovery in Services, IEEE Internet Computing, vol. 13, no. 2, pp. 88-91.

[8]. Semih Cetin, N. IlkerAltintas, HalitOguztuzun, Ali H. Dogru, OzgurTufekci, and Selma Suloglu,(2007) A Integration-Based Strategy for Migration to Service-Oriented Computing, Proc. IEEE Int'l Conf. Pervasive Services

[9]. McIlraith SA, Son TC, Zeng H, (2001),Semantic Web services. IEEE Intell Sys 16(2):46.

[10]. Chakraborty D, Perich F, Joshi A, Finin T, Yesha Y, (October 2002), A reactive service composition architecture for pervasive computing environments. In: Proceedings of the 7th personal wireless communications conference, Singapore, pp 53–62.

[11]. Chakraborty D, Perich F, Avancha S, Joshi A, (October 2001), DReggie: a smart service discovery technique for e-commerce applications. In: Proceedings of the workshop at the 20th symposium on reliable distributed systems, New Orleans.

[12]. Bussler C, Fensel D, Maedche A, (2002), A conceptual architecture for Semantic Web enabled Web services. SIGMOD Rec 31(4):24–29.

[13]. Cardoso J, Sheth A, (2002), Semantic e-workflow composition, Technical report, LSDIS Lab, Computer Science, University of Georgia.

[14]. http://www.w3schools.com.

[15]. A book Web Services an introduction by B V Kumar and S V Subrahmanya.

[16]. A book Data Mining introductory and advanced topics by Margaret H. Dunham.