International Journal of Soft Computing 10 (2): 157-162, 2015
ISSN: 1816-9503

Accuracy of Learning Data Through New Breed of Web-Based Applications Using Syntactic Approach

[1]M. Akila Rani and [2]G. Shanthi
[1]Department of Computer Science and Engineering,
N.P.R. College of Engineering and Technology, Natham-624401, Tamil Nadu, India
[2]Department of Computer Science and Engineering,
PSNA College of Engineering and Technology, Natham-624401, Tamil Nadu, India

Abstract: A script from different website forms a service mashup and its main characteristics are aggregating the data and combining them to make a visualizable outputs. A web service is used to exchange data between application and systems with a collection of open protocols. Through KDS (Knowledge Discovery in Services) on an open source of services from the internet we visually represent mashup to understand and predict the information in more accurate precision. In KDS according to the use we classify the matching technique from local to global, various complementary domains, merging more than one to same domain. The matching technique is classified synthetically into three kinds of layer they are granularity, kind of input layer, techniques layer. Here, comes the matching technique falls under element level matching technique which analyses entities or instances in isolation also ignoring their relations with other entities or their instances. The input interpretation layer chooses the syntactic technique with regard to its sole structure. The kind of input layer given to it in the form of terminological where strings found in the ontology descriptions. String based technique under element level are followed with more similar string, the more likely they are to denote the same concepts. Using distance function maps a pair of strings to a real number. This study introduces a process for KDS to assess the ability to predict service mashups using a combination of inputs and outputs.

Key words: KDS, service mashup, matching technique, pain, isolation

INTRODUCTION

The term Web Data Mining is a technique used to skulk through various web properties to gather required facts which enables an individual or a company to promote business, understanding marketing undercurrents, new raises floating on the internet, etc. There is a growing trend among companies, administrations and individuals alike to collect information through web data mining (Rocco *et al*., 2005) to apply that information in their best interest. A mashup, in web development, is a web page or web application that uses content from more than one basis to create a single new examination displayed in a single graphical interface. For example, we could combine the names and photographs of the library branches with a Google map to create a map mashup. It implies easy, wild integration, frequently using open application programming interfaces (open API) and data bases to produce enriched results that were not necessarily the original reason for producing the raw source data. In recent American English parlance, it can refer to audio with rhythms where people seamlessly combine audio from one song with the vocal track from another thereby mashing them together to create something new. Knowledge discovery in databases (Rocco *et al*., 2005) pursues new information in some application domain. It is defined as the nontrivial process of recognizing valid, novel, hypothetically useful and ultimately logical patterns in data. The process simplifies to no database sources of facts, although, it highlights databases as a primary source of data. It consists of many steps (one of them is DM), each struggling to complete a specific discovery task and each accomplished by the application of a discovery method. Knowledge discovery (Rocco *et al*., 2005) concerns the complete knowledge abstraction process including how data are stored and accessed, how to use competent and accessible algorithms to examine massive datasets how to understand and visualize the results and how to model and support the communication between human and machine. It also concerns support for erudition and examining the application domain.

Corresponding Author: M. Akila Rani, Department of Computer Science and Engineering,
N.P.R. College of Engineering and Technology, Natham-624401, Tamil Nadu, India

EXISTING SYSTEMS

Semantic matchmaking as logic approach: Matchmaking ascends when supply and demand meet in an electronic marketplace or when mediators search for a web service to achieve some job or even when employing agencies match curricula and job profiles. In such open environments, the objective of a Matchmaking Method is to discover best available proposals to a given request. We discourse the problem of matchmaking from a knowledge illustration perspective with a formalization based on description logics. We devise concept abduction and concept contraction as non-monotonic inferences in description logics suitable for modeling matchmaking in a logical framework and prove some related difficulty results. We also present reasonable algorithms for semantic matchmaking (Blake, 2009) based on the devised inferences and prove that they obey to some common sense properties.

Similarity measures and information retrieval techniques: It is chastely logic-based matching may have counter intuitive outcomes. It also permit for more springy methods of measuring service similarity. The IR and similarity-based methods are faultless candidates to learn the data more correctness. For example, linguistic semantics (Rao and Su, 2005) (Word Net similarity in Fig. 1), TF-IDF. Logic is fair one component of relevance. Such approaches capturesome other components.

Graph based approach: A service is characterized as a DAG (Blake, 2009) and their nodes-individuals of ideas. The arcs roles between individuals. The main idea in the graph based method is matched as a structural where two services explanations match if they have the same edifice and the corresponding nodes matches this is the basic concept followed in graph approach where a sample is given in Fig. 2.

Indirect graph-based matching: The indirect graph created matching under take a complex workflow compositions wherein "service chains" in the greenest case. The service chain creation rules are tracked:

- The inputs of each convoluted service match either the request inputs or the outputs of the preceding service in the chain
- Each output of the request is matched beside an output of the most recent service in the chain

A corresponding approach for discovery of complex service (Blake and Nowlan, 2008) workflow but applied through logic resolution. The central idea backward-chaining goal-driven reasoning way are initial from services that match the request outputs (but not its inputs), we recursively attempt to link them given in Fig. 3 with other services until we discover a service with all its inputs coordinated to the inputs given by the request. It also inherent support by reason programming tools (Prolog).

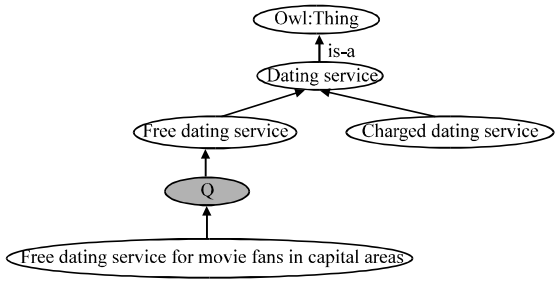

Fig. 1: Word net similarity

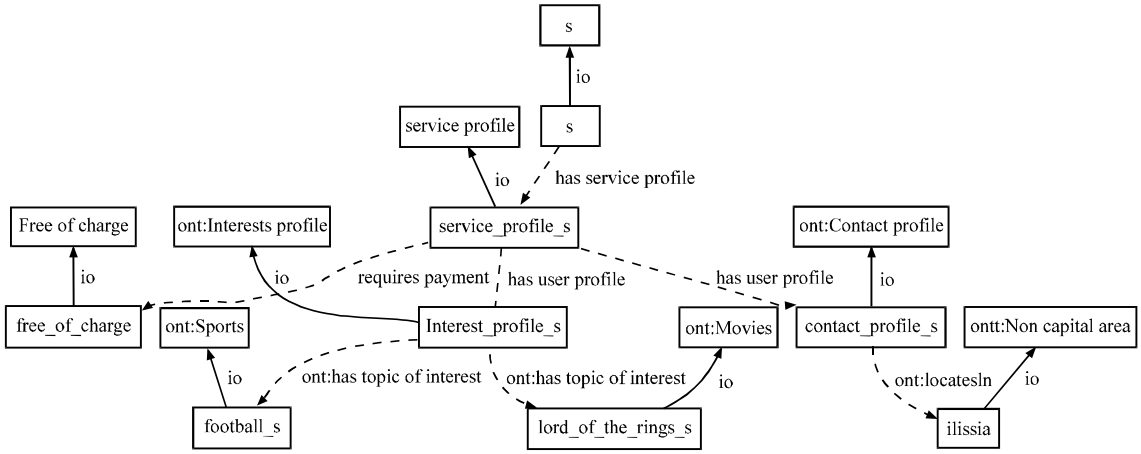

Fig. 2: Graph based approach

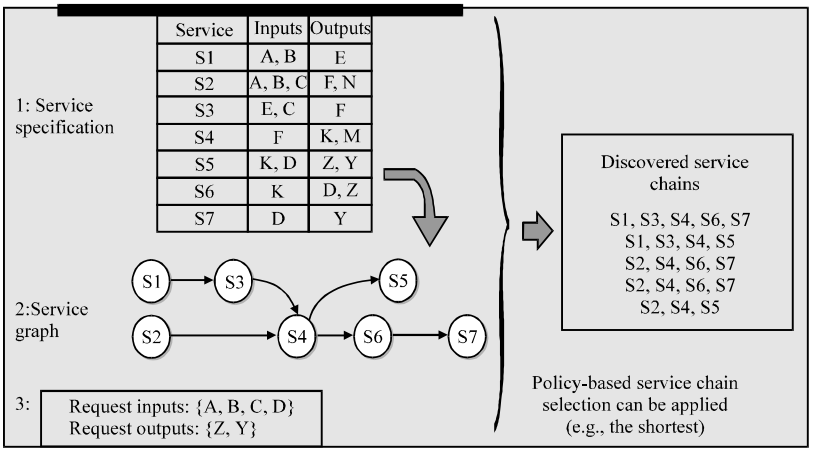

Fig. 3: Indirect graph-based matching

PROPOSED SYSTEM

In order to increase the learning accuracy and also to diminish the computational cost select a particular case illustrations from a database which contain a great amount of data. This study aim is to select a sample from the unique database as a subset and treated as training set. A new mechanism accuracy learning when associated with the existing random selection, this categorizes the maximum uncertain data from the original database.

In this study, we measure the ability to predict service mashups (Blake and Nowlan, 2008) using a combination of inputs and outputs. Also, we plan to summative the most useful message parts to develop core objects for new web service developers. By changing the number of categories and clusters, the KDS (Blake, 2009) algorithms realize higher precision. Also, the application is the formation of a distributed web services development situation that leverages the knowledge of existing services. Web service developers may, in real time, be suggested web services that might be consumed in the different applications that they develop. The plan would be the integration of the method for use as a front-end to the emerging service mashup editors and visualization environments. As such, it would be important to estimate the approach on languages other than English. By integrating the methodology to the front-end of service mashup processors, it can suggest potentially useful mashup and consequently the editors and visualization tools can portray the output.

Levenshtein algorithm: The Levenshtein algorithm (also called edit-distance) analyses the least number of

		m	e	i	l	e	n	s	t	e	i	n
	0	1	2	3	4	5	6	7	8	9	10	11
l	1	1	2	3	3	4	5	6	7	8	9	10
e	2	2	1	2	3	3	4	5	6	7	8	9
v	3	3	2	2	3	4	4	5	6	7	8	9
e	4	4	3	3	3	3	4	5	6	6	7	8
n	5	5	4	4	4	4	3	4	5	6	7	7
s	6	6	5	5	5	5	4	3	4	5	6	7
h	7	7	6	6	6	6	5	4	4	5	6	7
t	8	8	7	7	7	7	6	5	4	5	6	7
e	9	9	8	8	8	7	7	6	5	4	5	6
i	10	10	9	8	9	8	8	7	6	5	4	5
n	11	11	10	9	9	9	8	8	7	6	5	4

Fig. 4: Levenshtein approach

correct operations that are essential to modify one string to obtain another string. The most shared way of computing this is by the dynamic programming approach. A matrix is initialized calculating in the (m, n) cell the Levenshtein distance between the m-character prefix of one with the n-prefix of the additional word. The matrix can be occupied from the upper left to the lower right corner. Each jump horizontally or vertically corresponds to an insert or a delete, respectively. The cost is normally set to 1 for each of the operations. The diagonal jump can cost either one, if the two characters in the row and column do not match or 0, if they do. Each cell always minimizes the cost locally. This way the number in the lower right corner is the Levenshtein distance between both words. Here, is an example that features the 1 comparison of meilenstein and "levenshtein".

There are two possible paths through the matrix given in Fig. 4 that actually produce the least cost solution. Namely:

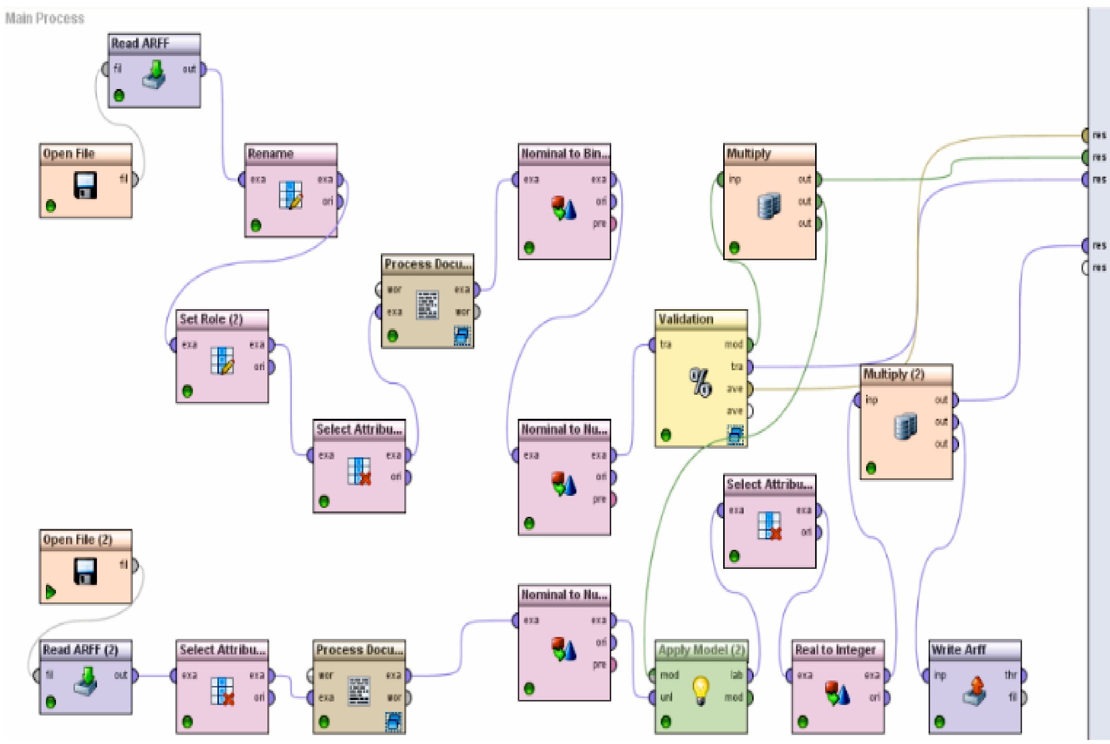

Fig. 5: Predict the cluster using rapid miner

l	e		v	e	n	s	h	t	e	i	n		l	e	v		e	n	s	h	t	e	i	n
o	=	+	o	=	=	=	=	=	=	=	=	or	o	=	o	+	=	=	=	=	=	=	=	=
m	e	i	l	e	n	s		t	e	i	n		m	e	i	l	e	n	s		t	e	i	n

"=" Match; "o" Substitution; "+" Insertion; "-" Deletion First step in the approach to predicting a service mashup (Blake and Nowlan, 2008). This approach combines syntactical methods with human naming tendencies to increase the probability of the messages having equivalent meanings. After finding similarity between web service messages, we place services with similar messages into clusters. Services can be associated to more than one cluster but later they will have be attached to only one category. In the final phase, we attempt to predict the clusters (Blake and Nowlan, 2008) that will add value to the end users. This could be done through rapid miner which is given through the Fig. 5 (Benatallah *et al*., 2005), clustering is an unsupervised data mining technique where the task is to capture the possible natural groupings between different objects represented in the data. It works by defining a "closeness" metric for comparing two objects and then grouping all objects with a similar closeness. There are really two main classes of clustering methods: top-down or bottom-up.

In a top-down approach, we will (arbitrarily) decide how many clusters the data would form. Then, we put the objects into a particular cluster if they are within a certain distance or closeness to the center of that cluster. The popular k-means Clustering algorithm is a good example of a top down approach. There are variations of this top down approach where we would not need to prespecify the number of clusters, for example the Density Based Scan Method (DB Scan) relies on computing object density to naturally detect groups.

In a bottom-up approach which is explained in Fig. 6, we start by defining a distance metric. Then, find two objects which are within this closeness and merge them into one group. Then, we look for the next pair of objects which meet the criterion, merge them and so on. Hierarchical or agglomerative clustering falls into this bottom-up approach.

Data-Based Model filtering helps you create mining models (Benatallah *et al*., 2005) use subsets of data in a mining structure. Filtering gives you flexibility when you design your mining structures and data sources because you can create a single mining structure, based on a comprehensive data source view.

We can then create filters to use only a part of that data for training and testing a variety of models, instead of building a different structure and related model for each subset of data.

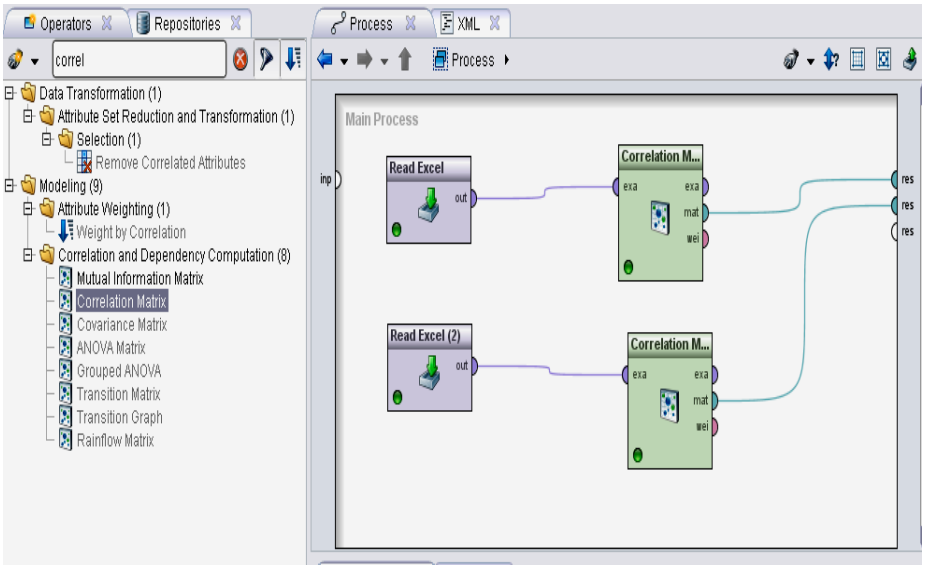

Fig. 6: Filtering using rapid miner

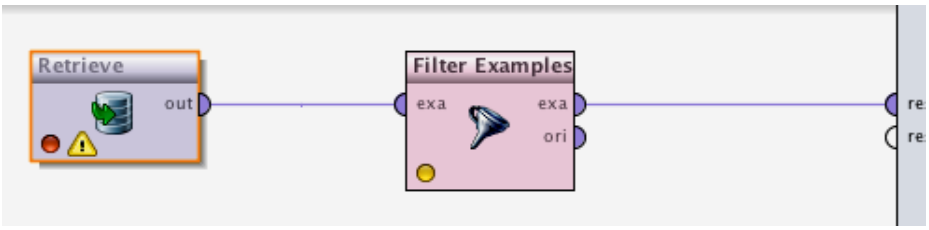

Fig. 7: Using models to group

For example, you define the data source view on the customers table and related tables. Next, you define a single mining structure that includes all the fields you need. Finally, you create a model that is filtered on a particular customer attribute such as region. You can then easily make a copy of that model and change just the filter condition to generate a new model based on a different region. Some real-life scenarios where you might benefit from this feature include the following:

- Creating separate models for discrete values such as gender, regions and so forth. For example, a clothing store might use customer demographics to build separate models by gender even though the sales data comes from a single data source for all customers
- Experimenting with models by creating and then testing multiple groupings of the same data such as ages 20-30 vs. 20-40 vs. 20-25 (Fig. 7) (Benatallah *et al*., 2005)
- Specifying complex filters on nested table contents such as requiring that a case be included in the model only if the customer has purchased at least two of a particular item

CONCLUSION

In this study, we are able to enable efficient Web Service discovery on large scale, we do not only need access to a large number of publicly available services, we also want these services to be syntactically annotated. Within this study, we present the demonstration using rapid miner a tool that enables the easy annotation of Web APIs, i.e., both WSDL and REST based services. With this tool we are able to process for KDS (Blake, 2009) to assess the ability to predict service mashups using a combination of inputs and outputs.

REFERENCES

Benatallah, B., M. Dumas and Q.Z. Sheng, 2005. Facilitating the rapid development and scalable orchestration of composite web services. Distrib. Parallel Databases, 17: 5-37.

Blake, M.B. and M.F. Nowlan, 2008. Predicting service mashup candidates using enhanced syntactical message management. Proceedings of the IEEE International Conference on Services Computing, Volume 1, July 7-11, 2008, Honolulu, HI, pp: 229-236.

Blake, M.B., 2009. Knowledge discovery in services. IEEE Internet Comput., 13: 88-91.

Rao, J. and X. Su, 2005. A Survey of Automated Web Service Composition Methods. In: Semantic Web Services and Web Process Composition, Cardoso, J. and A. Sheth (Eds.). LNCS. Vol., 3387, Springer, Berlin, Heidelberg, pp: 43-54.

Rocco, D., J. Caverlee, L. Liu and T. Critchlow, 2005. Domain-specific web service discovery with service class descriptions. Proceedings of the IEEE International Conference on Web Services, July 11-15, 2005, Orlando, Florida, USA., pp: 481-488.