IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 10, Issue.2, February 2022 E-ISSN: 2347-2693

# Automatic Composition of Machine Learning Models as Web Services across Data Sets

**Mohan H.G.[1*], Nandish M.[2], Devaraj F.V.[3]**

[1,2]Department of Computer Science and Engineering, JNNCE, Shimoga, India
[3]Department of Computer Science and Engineering, PESITM, Shimoga, India

[*]*Corresponding Author: mohan.hg0408@gmail.com, Tel.: +91-9620866080*

DOI: https://doi.org/10.26438/ijcse/v10i2.710 | Available online at: www.ijcseonline.org

Received: 28/Feb/2022, Accepted: 08/Feb/2022, Published: 28/Feb/2022

***Abstract*—** Machine Learning (ML) is a field of Artificial Intelligence, which applies the principles of statistics to predict the outcome of process by utilizing the historical data. Each ML model is built on a specific Data Set and programming language, however the knowledge obtained by the model is restricted to that particular Dataset. Web Service Composition (WSC) process of combining the available web services (WS) and arriving at a new web service based on the required inputs and preconditions. The paper presents an approach to represent a ML model as web service and automatically composing them with other ML models to utilize the knowledge gained on different data sets. The MLWSC approach consists of two stages. In stage one, semantic networks are created among the ML services to form a network. In stage two, MLWSC compositions are constructed based on the requirement of the user.

***Keywords*—** Web Service Composition, Machine Learning, Data Sets.

## I. INTRODUCTION

Machine Learning is the bussing field of computer science over the last decade paving way for efficient utilization of computational capabilities of modern machines to either classify or predict the outcome [1]. The ML models are built on the statistical methods on the underlying datasets, the models are trained, tested on the dataset and validated on some percentage of the dataset rows. The model improves its efficiency in these learning's. However in all the applications the model is built on a single dataset and either implemented in python or R or java languages. The knowledge gained by the model is restricted to the underlying data set used. It cannot be used on other datasets. The programming language acts a hindrance to integrate different models developed in different programming languages. These restrictions pave way for the proposed method which describes a ML model as Web service and combine them through a composition method to effectively integrate the models to obtain the desired output.

A Web service is software based application which is described uniquely by its URI, its interfaces and bindings have the capability of identification, discovered and described on its own by XML configurations, it supports the direct communication with other set of software applications. Web services are based on XML messages over internet based communication and protocols [2]. Web services provide the facility to communicate and coordinate with other computing platforms, applications, languages and business entities. Web services acts like a black box, without having to bother about the internal working, hardware, platform or programming language. Its interfaces are sufficient to interact with other entities. The composition of the WS is a challenging task given the fact that each WS has its own inputs and outputs. The pre-conditions and effects of a service can also be different [3]. Thus the effective ability of service lies in selecting, integrating the services at run time is a challenging task towards the compositions of web services.

In this paper we consider the problem of representing Machine Learning models as Web services by considering its inputs, outputs and Data sets. The ML model is encapsulated to form a web service by specifying the interfaces to the WS in WSDL file [4]. Based on the user requests the MLWS are combined together by the composition algorithm considering the ranking among the ML models based on their accuracy levels [5].

The rest of the paper is organized as follows. The work related to MLWS is explained in section II. Proposed way of encapsulating a ML model in a WS is explained in section III. Experimental results are presented in section III. Concluding remarks are mentioned in section IV.

## II. RELATED WORK

The work presented in [6] provides a method called MLS-PLAN algorithm to compose new machine learning services. It discusses a use case of automated machine learning in real-world scenario. The research presented in [7] deals with the definition of program software that provides an opportunity to solve the problem of machine separation data learning methods for input in the context of

 

various learning areas. A list of such activities may include: suspicious identification state contractors for cooperating with suppliers, predicting the execution of a government contract, predicting the revocation of a license credit unions and insurance companies, etc. The application allows the user to build several basic hardware configurations learning models to solve problems in stages. As part of the research, the functional requirements of the product being processed are presented, architecture developed, design and evaluation of the proposed process for creating a machine learning model done. The use of the app will allow you to create songs for machine learning models in easy-to-use mode to increase the accuracy of the partition problem. The use of proposed tools enhances the accuracy of problem solving data input in context of various topics using the formation of machine learning mechanisms. The new trend of machine learning as a service is presented in [8].

## III. MACHINE LEARNING MODEL AS WEB SERVICE

A Machine Learning Model can be built for Classification, Regression, Forecasting, Clustering or Dimension Reduction problems. They have their own set of input parameters and outputs (effects). Each model runs overs a specified data set. These data sets contain features on which the model is built for the required application. The knowledge gained such as parameter tuning, optimization values by the model is persistently stored in a knowledge bank and can be reused on different datasets. The ML Model is described in Figure 1. Each of these models is implemented either in Java, Python or R languages.

The Machine learning model is encapsulated in a web service by providing interfaces. The ML web service is described using the WSDL file; the WS will now contain the preconditions and effects data that are required by the underlying ML model. The WSDL file acts as the contract for the web service. The embedding layer embeds service description into ML models. The MLWS is as shown in Figure 2. Web services are stored on an API sharing platform which makes these services discoverable [9].

### A. *Semantic Links (SL)*

The Machine Learning Web services (MLWS) functionalities are described using their inputs, output parameters and preconditions and effects. The MLWS are self-describing in nature. In MLWS proposed method, inputs are identified as *I_s* and *O_s* as output parameters of MLWS, *mlws*. A SL is expressed as triplet as used in (1) here, *mlwsi* & *mlwsj* are MLWS. The semantic similarities are calculated based on the SL between the parameters of services.

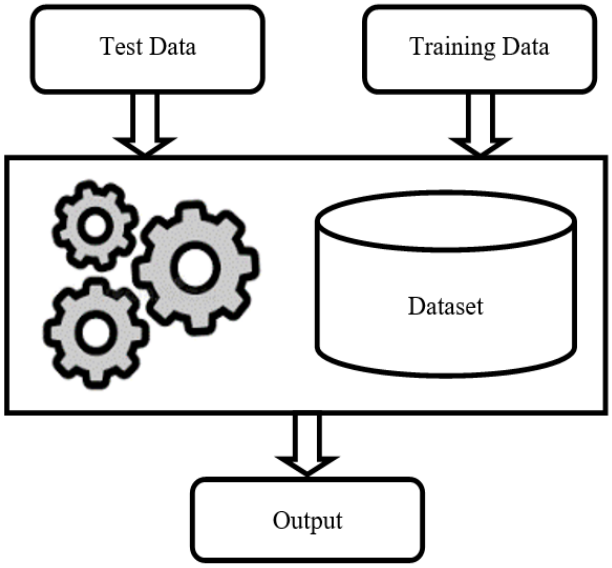


Figure 1 : Machine Learning Model

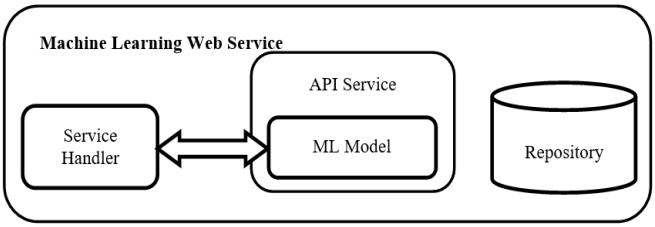


Figure 2 : ML Model as a Web Service

The MLWS similarities are identified based on a matching calculation function among the two ML models. The matching function *SLij*, compare the two ML based on their input and outputs of the underlying model, models will be matched based on functionality of the machine learning model rather than on the method that is being used and it also do not depend on data set used [10].

$$SLij = <mlwsi, SimT(O\_si, I\_sj), mlwsj> \quad (1)$$

The MLWS composition process involves retrieving of SLs, SLij among an output *O_si* of MLWS *mlwsi* and input *I_sj* of other MLWS, *mlwsj* before causal laws among effects, *Emlwsi* of *mlwsi* and preconditions, *Pmlwsj* of *mlwsj* which means, *Emlwsi* imply *Pmlwsj*. Therefore, we can establish that *mlwsi* and *mlwsj* are can be partially hooked based on a similarity matching formula, *SimlarT*, specifying a data flow. It means the presence of a SLij implies that *mlwsj* succeeds *mlwsi* and the output of *mlwsi* can be fed as an input to *mlwsj* without any interleaved services between *O_si* and *I_sj*.

### B. *Service Link Network (SLN) Generation*

With a given set of MLWS, the SLN is constructed with an aim to generate a network of MLWS based on their SLs and constraints defined as causal laws [11]. A SLN is represented using a 4-tuple notation <S, C, δ, λ> here:

- S is a set of MLWS.
- C is a set of conflicts between MLWS.

 

- δ is S × {E,P,U,I} × S, indicates SLs between MLWS, evaluated using *SimlarT* [12].
- λ is C × {E,P,U,I}× S, indicates conflicting MLWS in S evaluated by *SimlarT*.

The MLWS which provide same functionality and having same input and output parameters are said to be conflicting with each other [13]. The MLWS that produces same output are also said to be in conflict with each other since each of them can produce the same output. The MLWS are said to be in Precondition conflicts if they have incompatible preconditions, during execution only one among them will be selected.

The steps in SLN generation are

1. ***foreach*** *pair of MLWS (mlwsi, mlwsj)*
2. *Compute SimlarT (O_si, I_sj)*
3. ***if*** *input of MLWS mlwsi and mlssj are same* ***then***
4. *{input conflicts} = {input conflicts} U {mlwsi , mlssj}*
5. ***if*** *input of MLWS mlwsi and mlwsj have complimentary pre-conditions* ***then***
6. *{pre-condition conflicts} = {pre-condition conflicts} U {mlwsi , mlssj}*
7. ***if*** *input of MLWS mlwsk is satisfied by both mlwsi and mlwsj* ***then***
8. *{conflicting service on mlwsk} = {conflicting service on mlwsk} U {mlwsi , mlwsj}*

### C. *MLWS Acyclic Dependency Path*

SLN is extended by adding intermediate operators to generate a process structure, MLWS Acyclic Dependency Path (MADP), represented as a 6-tuple <N, S, G, E, start, end, type> where [14]:

- *N* is a set of nodes which can be S or G.
- *S* is a set of MLWS.
- *G* is a set of gateways.
- *E* is $N \times N$ the set of edges between MLWS.
- *start* is start service, for each node $n \varepsilon N$ have a path from start to *n*.
- *end* is end service where execution terminates.
- *type* : G →{SPLIT, MERGE} is a function to labels a node.

The steps in constructing MADP from SLN are

1. *For every node in the SLN*
2. *If a node is having many outgoing nodes then*
   *A Gateway is add after the node*
3. *If all outgoing nodes are in conflict sets with each other then*
   *The type of the gateway is set to 'SPLIT'*
4. *If few or all incoming nodes are not in set of conflict then*
   *The gateway type is set to 'MERGE'*
5. *If few outgoing nodes are in conflicts then*
6. *Create a node 'MERGE' gateway for the conflicting nodes*

### D. *MLWS Composition*

The MADP is converted into structured composition which becomes executable. Structured processes are created to represent MLWS compositions [15]. The structured processes are encoded in any executable language, which adds to its benefits. The composition C is represented as (2).

$$C ::= \text{MERGE}\{C,...,C\} \mid \text{SPLIT}\{C,...,C\} \mid C \rightarrow C \mid s \quad (2)$$

Here '*s'* represents an Atomic statement which is execution of a MLWS, operator '→' indicates sequential composition, operator 'SPLIT' indicates parallel execution, 'MERGE' operator specifies convergence to wait till completion of all of its parameters services to continue. The SLNs are acyclic, the constructed service compositions will not have loops. A MADP is structured if gateway SPLIT and MERGE exists in pairs. However, MADP graphs are unstructured in general.

The terms predecessor, immediate predecessor, successor and immediate successor nodes are used to derive flow path. The MADP will not be having start and end nodes, so a START and END node is added and an edge is added from all final nodes to END node.

The input to composition task is a MADP and a node $x \in S \cup G$. This node is processed in the task, it is recursive and exits once processing of all nodes of MADP are completed resulting in a structured composition.

Steps in Composition Process are

1. *Create a START node.*
2. *For the given input parameters of the MLWS task, find the services with same input parameters, create an edge from START node to each such services selected.*
3. *Create an END node.*
4. *For the given output parameters of the task, find the services with same output parameters, create an edge from each such services selected to END node.*
5. *Using MLWS Acyclic Dependency Path, find the paths from nodes selected in step-2 to nodes selected in step-4.*
6. *If multiple paths are available in step-5, then evaluate the performance of each MLWS involved, based the correlations of execution time, accuracy, precision, and recall rates.*
7. *Select the best path and execute the MLWS to obtain the required output.*

## IV. RESULTS AND DISCUSSION

We used 10 machine learning methods on the proposed MLWS composition method which included Linear Regression, Logistic Regression, Decision Tree, SVM Algorithm, Naive Bayes Algorithm, AdaBoost, CNN, KNN Algorithm, K-Means, Random Forest Algorithm along with their required service datasets the details are mentioned in Table 1. Each of these models are encapsulated in a web service identified by their bindings. Each ML model is ran using their specified dataset. The composition method is applied on a page ranking and recommendation systems.

 

The composition algorithm is applied on the page ranking system by using the Acyclic Dependency Path generated with the given MLWS and their datasets. Given the inputs and outputs of the page ranking system. The composition system will find the sequence of MLWS to be executed to produce the required result. The method selects the Decision Tree followed by KNN and Random Forest which is having the less execution and accuracy compared to all other paths in dependency graph. The experiment was made on i5. Each model ran on its own dedicated instances (2 CPUs, 4GB RAM). The composition process was successful in composition the MLWS.

Table 1. Different ML Models as MLWS

| Model used as MLWS | Time Taken | Accuracy |
|---|---|---|
| Linear Regression | 45s | 83.22% |
| Logistic Regression | 50s | 87.55% |
| Decision Tree | 92s | 89.63% |
| SVM | 74s | 91.32% |
| Naive Bayes | 57s | 91.86% |
| AdaBoost | 67s | 88.63% |
| CNN | 63s | 89.79% |
| KNN Algorithm | 48s | 86.59% |
| K-Means | 59s | 94.36% |
| Random Forest | 53s | 93.45% |
| Composition of Decision Tree followed by KNN and Random Forest | 350s | 85.82% |

## V. CONCLUSION AND FUTURE SCOPE

In this paper we have presented a method to convert the Machine Learning Model into an Executable Web Service, These MLWS process the quality of the web services such that they are identified by their bindings and become discoverable for the composition process. Each of these MLWS are identified by their inputs and output parameters and preconditions and effects. Using the casual laws the MLWS are semantically liked to each other. These models form a pipeline for execution when more than one MLWS can be selected, we can further include a correlation among those and select one with a better performance. The main benefit of MLWS composition is that it avoid recreating a ML model, instead the existing ML models can be plugged to each other to obtain the required objective. In future the proposed method can be extended to non-ML computational models. The registry based functionality can be added to form a repository so that the MLWS can be added and remove dynamically at run time.

## REFERENCES


[1] Rik Eshuis, Freddy Lecue, and Nikolay Mehandjiev, *"Flexible Construction of Complex Service Compositions from Reusable Semantic Knowledge"*, In the Proceedings of IEEE 19th International Conference on Web Services, pp.**631-632**, **2012**.

[2] Paolo Traverso and Marco Pistore, *"Automated Composition of Semantic Web Services into Executable Processes*", In the Proceedings Third International Semantic Web Conference, Hiroshima, Japan, pp.**380-394**, **2004**.

[3] Mohan H G, Chetan K R, *"Semantic Based Automatic Web Service Composition"*, International Journal of Applied Research and Studies, Vol.**3**, Issue.**6**, **2014**.

[4] Mohan H G and Devaraj F V, *"A Survey on Semantic Based Automatic Web Service Compositions"*, European Journal of Advances in Engineering and Technology, pp.**73-79**, **2014**.

[5] Chatti Subbalakshmi, Rishi Sayal , H. S. Saini, *"S-REST: A design of Secured Protocol for Implementation of RESTful Webservices"*. International Journal of Computer Sciences and Engineering Vol.**7(1)**, Jan **2019**.

[6] Felix Mohr, Marcel Wever, Eyke Hullermeier, Amin Faez, *"Towards the Automated Composition of Machine Learning Services"*, In the IEEE International Conference on Services Computing, **2018**.

[7] Igor Lavrov and Jenny Domashova, "*Constructor of compositions of machine learning models for solving classification problems"*, In the Postproceedings of the 10th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA **2019**.

[8] Kailash Chander Bhardwaj and R K Sharma, *"Machine Learning In Efficient And Effective Web Service Discovery"*, Journal of Web Engineering, Vol. **14**, No.3&4, pp.**196-214**, **2015**.

[9] Y. Yang, X. Li, Z. Liu, and W. Ke, *"RM2PT: A tool for automated prototype generation from requirements model"*, In the Proceedings of International Conference on Software Engineering, May **2019**.

[10] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, *"Quality driven web services composition"*, In the Proceedings of the 12th international conference on World Wide Web. ACM, pp.**411–421**, **2003**.

[11] F. Mohr, A. Jungmann, and H. Kleine Buning, *"Automated online service composition"*, In the Proceedings of the International Conference on Services Computing, IEEE, pp.**57-64**, **2015**.

[12] Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, *"Automatic Composition of e-services that export their behavior"*, In the Proceedings of the International Conference on Service-Oriented Computing. Springer, pp.**43-58**, **2003**.

[13] Freddy Lécué and Alain Léger, *"Semantic Web Service Composition through a Match making of Domain"*, In the Proceedings of IEEE 4th European Conference on Web Services, pp.**171-180**, **2006**.

[14] Yilong Yang, Nafees Qamar, Peng Liu, Katarina Grolinger, Weiru Wang, Zhi Li, Zhifang Liao, *"ServeNet: A Deep Neural Network for Web Services Classification"*, In the 12th International Conferences on Web Services, China, Oct. **2020**.

[15] Jing Zhang, Yang Chen, Yilong Yang, Changran Lei, Deqiang Wang, *"ServeNet-LT: A Normalized Multi-head Deep Neural Network for Long-tailed Web Services Classification"*, In the IEEE International Conferences on Web Services, Sep. **2021**.


 