A Deep Learning Approach for Long Term QoS-Compliant Service Composition

Hamza Labbaci[1,2], Brahim Medjahed[2(✉)], and Youcef Aklouf[1]

[1] USTHB University, Algiers, Algeria
hlabbaci@umich.edu, yaklouf@usthb.dz
[2] University of Michigan - Dearborn, Dearborn, USA
brahim@umich.edu

Abstract. In this paper, we propose a deep learning approach for long-term Quality of Service (QoS)-based service composition. Existing techniques for quality-aware service composition mostly focus on static QoS values observed during composition time. They do not consider potential QoS fluctuations in the long run when selecting services for composition or substitution. Our approach uses deep recurrent Long Short Term Memories (LSTMs) to forecast future QoS. The predicted QoS values are used to accurately recommend components and substitutes in long-term service compositions. Experiments show promising results compared to existing QoS prediction techniques.

Keywords: Service composition · Substitution · Quality of Service (QoS) · Deep learning · LSTMs

1 Introduction

During the last decade, many organizations embraced service-oriented computing technologies, seeking better visibility and more market opportunities. Web services (APIs) with complementary functionalities (called *components*) collaborate as part of the same *service composition* to provide value-added services [5,7]. The success and longevity of collaborations in a service composition strongly depend on the ability of the different components to maintain *long-term* Quality of Service (QoS) requirements [4]. Developing long-term compositions raises the challenge of selecting components that satisfy QoS requirements over long time periods. Such selection implies predicting long-term QoS trends (i.e., QoS during a long period). The main challenge related to forecasting long-term QoS is that QoS values may fluctuate in the future.

We identify three advantages for leveraging long-term QoS trends during composition. First, developers rely on predicted QoS to accurately select the best services that are likely to fulfill composition requirements over a long time period. This caters for durable partnerships among component services. Second, component services undergo several changes during their lifespan (e.g., a service going out of business) that may lead to breaking contracts between composite and component services. Developers will then be able to substitute components by services with comparable long-term QoS. Third, service providers

M. Maximilien et al. (Eds.): ICSOC 2017, LNCS 10601, pp. 287–294, 2017.
https://doi.org/10.1007/978-3-319-69035-3_20

(e.g., cloud providers) may rely on QoS prediction for better resources management and workload balancing. For instance, they may harness more server resources (processor and memory) during peak periods. Accurate QoS prediction allows service providers to adjust their cloud resources to satisfy users' demands as accurately as possible in the future. This reduces contract disruption between long-term composite services and their components because of QoS violations.

Several techniques for QoS-based service composition have been proposed in the literature [9,11,12]. However, they mostly focus on static QoS values observed during composition time. In practice, QoS varies over time; such future variations need to be taken into consideration while designing composite services. Unlike current techniques, we propose a deep learning approach for long-term service compositions. We use deep recurrent Long Short Term Memories (LSTMs) [3] to predict long-term QoS trends. Such predictions cater for selecting best Web services that satisfy QoS-related composition requirements in the long run. To the best of our knowledge, this is the first work that uses deep learning and particularly deep recurrent LSTMs for long-term QoS-aware service composition and substitution.

The rest of the paper is organized as follows. Section 2 describes our deep learning approach for QoS prediction. Section 3 presents our technique for leveraging the predicted QoS during composition and substitution times. Section 4 discusses the experimental study. Section 5 reviews related work. Section 6 concludes the paper.

2 Using Deep Learning for QoS Prediction

The aim of the proposed approach is to assist developers in designing composite services and substituting components while considering long-term QoS trends. The composition middleware implements techniques to identify, orchestrate, and substitute component services. The way component and substitute services are selected by the middleware is out of the scope of this paper. Our approach augments any existing composition and substitution technique such as the ones proposed in [5] with long-term QoS prediction and compliance capabilities.

2.1 QoS Composition Requirements

Developers provide two kinds of composition requirements to the composition middleware: middleware-specific requirements such as functional and semantic features (out of the scope of this paper) and *QoS requirements*. For each QoS metric, developers specify *preferred* and *acceptable* intervals. Figure 1 shows an example of QoS requirements for processing speed between $[T_0, T_1]$ (e.g., this year's summer season), $[T_1, T_2]$ (e.g. this year's fall season), ..., $[T_7, T_8]$. Gray and dark rectangles refer to acceptable and preferred QoS intervals, respectively. Figure 1 also shows the predicted computation speed $Q_1(t)$, $Q_2(t)$, and $Q_3(t)$ that three services S_1, S_2, and S_3 are likely to guarantee the next two years. Our approach compares the areas under the predicted curves with the area under the required QoS that are either bounded by the acceptable or preferred rectangles.

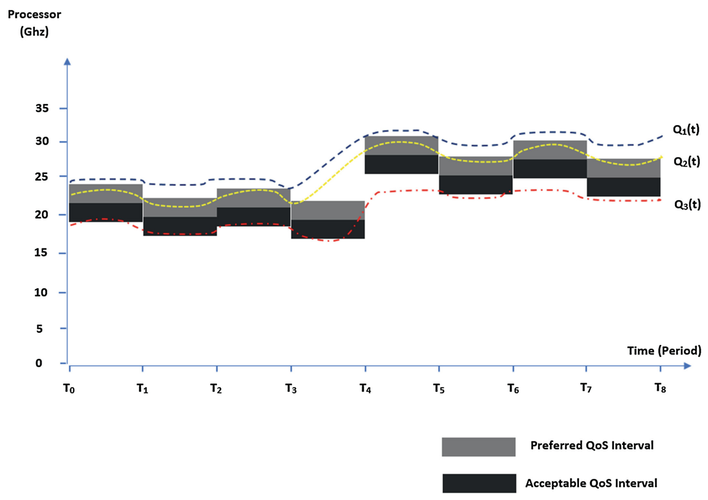

Fig. 1. Examples of QoS Prediction Trends

2.2 Predicting Long-Term QoS Trends

The *QoS Predictor (QP)* handles QoS prediction for all services and stores the predicted QoS curves in a *QoS Repository*. It has access to log files that contain QoS data observed over time for the different Web services in the system. The way QoS is monitored and obtained is out of the scope of this paper. [4] and [2] give details about QoS monitoring techniques.

QP learns from the history of observed QoS values to forecast long-term QoS trend. For that purpose, we train deep recurrent Long Short Term Memories (LSTMs) [3] with sequences of QoS values observed at different periods of time. LSTM is a particular type recurrent nets with the ability to avoid the long-term dependency problem. They can remember information for long periods of time. Unlike classic recurrent nets, LSTMs overcome very well the problem of vanishing gradient and can learn from size variable sequences of data. LSTM is trained with n raw data X_j. Each X_j is composed by an input Q_i and an output Q_{i+1} where Q_i and Q_{i+1} denote observed service QoS at times T_i and T_{i+1} respectively. The goal behind training the LSTM with sequences of QoS is to predict services future QoS values for a given period of time.

Predicting QoS trends is time consuming. To minimize the incurred overhead on the overall composition development (during component selection) and execution (during component substitution) times, we define two techniques for bootstrapping QoS prediction: *random* and *first-hit*. The *random* technique arbitrarily selects a service in the system during off-peak hours, and executes prediction algorithms for that service. This technique has the advantage of calculating predictions offline, with little or no impact on composition/substitution times. However, it may perform prediction for services that are never used in compositions/substitutions. The *first-hit* bootstrapping technique runs prediction algorithms for a service when it first participates in a composition/substitution.

In contrast to random bootstrapping, this technique carries out prediction only for services that are used in compositions or substitutions. Although the first-hit method is executed online (during composition or substitution), its overhead is reduced by limiting it to first-time composed or substituted services.

Once QoS prediction trends are generated, there is a need to keep them up-to-date as more QoS data is gathered. Performing prediction on more QoS data improves prediction accuracy. We define three event-based techniques for updating QoS predictions: *periodic*, *popularity-based*, and *change-based* QoS update techniques. The *periodic* technique updates prediction at the end of each time period T (e.g., at the end of each week). The *popularity-based* technique updates prediction whenever a service reaches a new popularity level. We say that a service is *popular* if it participated in N composition and substitution since the latest update. The *change-based* technique updates prediction for a given service whenever big changes, above a certain level C, are noticed in the QoS log of that service. Note that values of T, N, and C are selected by cloud providers and may be adjusted to deal with various environment conditions.

3 Long-Term QoS Compliance Checking

The composition middleware sends two types of recommendation requests to the *Long-Term QoS Compliance Checker (LQCC)*: composition and substitution requests. A composition recommendation request includes the ID of a potential component along with QoS requirements to LQCC. LQCC requests the QoS prediction trend for the component from the *QoS Predictor*. Then, it checks compatibility between the QoS prediction and QoS requirement intervals, and returns a composition recommendation to the middleware. A substitution recommendation request includes IDs of the services to substitute and potential substitutes to LQCC. LQCC obtains the QoS prediction trends for the component to substitute and potential substitute from the QoS Predictor. Then, it checks whether the two trends are close enough to each other and returns a substitution recommendation to LQCC.

3.1 Checking Compliance for Service Composition

We introduce two heuristics to check long-term QoS compliance for composition: *Conservative* and *Soft* heuristics. The *conservative* heuristic states that a service S is long-term QoS compliant with the developer's QoS requirement iff for each time interval $[T_i, T_{i+1}]$ in the prediction time interval $[\alpha, \beta]$, the area under the predicted QoS curve of S is greater than the area under the lower bound of the *preferred* QoS requirement curve, and is less than or equal the area under the higher bound of the *preferred* QoS requirement curve. The rational behind this heuristic is to make sure the component's QoS prediction curve remains within the preferred interval throughout the various time periods.

$$\int_{t_i}^{t_{i+1}} (q(t) - p_{lower}(t)) > \epsilon \; and \int_{t_i}^{t_{i+1}} (p_{higher}(t) - q(t)) > \epsilon$$

where ϵ ($\epsilon \geq 0$) is a composition compliance threshold, $q(t)$, $p_{higher}(t)$, and $p_{lower}(t)$ stand for the component's predicted QoS, the *preferred* higher bound QoS, and the preferred lower bound QoS respectively.

The *soft* composition compliance heuristic states that a component is long-term QoS compliant with the developer's QoS requirements iff the sum of the areas under the component's predicted QoS curves is superior to the sum of the areas under the lower bound of the *acceptable* QoS curves, and is less than or equals to the sum of the areas under the higher bound of the *preferred* QoS curves. The rationale behind this heuristic is to make sure that the overall component's QoS prediction curve is within any of the preferred or acceptable QoS intervals. The component's curve may fall outside the QoS requirement boundaries within a certain time period $[T_i, T_{i+1}]$ as long as there are other time periods that make up for the QoS loss in $[T_i, T_{i+1}]$.

$$(\sum_{i=\alpha}^{i=\beta-1} \int_{t_i}^{t_{i+1}} (q(t) - a_{lower}(t))) > \epsilon \ and \ (\sum_{i=\alpha}^{i=\beta-1} \int_{t_i}^{t_{i+1}} (p_{higher}(t) - q(t))) > \epsilon$$

where ϵ is a composition compliance threshold ($\epsilon \geq 0$), $q(t)$, $p_{higer}(t)$, and $a_{lower}(t)$ stand for the predicted QoS, the *preferred* QoS requirement upper bound, and the *acceptable* QoS requirement lower bound.

3.2 Checking Compliance for Service Substitution

Similarly to composition, we introduce two heuristics to check long-term QoS compliance for substitution: *Conservative* and *Soft* substitution heuristics. The *conservative* heuristic states that a service S is long-term QoS compliant with a potential candidate substitute C iff for each time interval $[T_i, T_{i+1}]$ in the prediction time interval $[\alpha, \beta]$, the difference between the two areas under the predicted QoS curves of S and C is less than or equal a threshold value ϵ ($\epsilon \geq 0$).

$$\int_{t_i}^{t_{i+1}} |(q_S(t) - q_C(t))| \leq \epsilon$$

where $q_S(t)$ and $q_C(t)$ stand for the predicted QoS of S and C respectively.

The *soft* heuristic states that a service S is long-term QoS compliant with a potential candidate substitute C iff the sum of the differences between the areas under the predicted QoS curves of S and C is less than or equal to the sum of the areas under the required QoS curves.

$$(\sum_{i=\alpha}^{i=\beta-1} \int_{t_i}^{t_{i+1}} |(q_S(t) - q_C(t))|) \leq \epsilon$$

where ϵ is a threshold such that $\epsilon \geq 0$, $q_S(t)$ and $q_C(t)$ stand for the predicted QoS of S and C, respectively.

4 Experimental Study

The goal of our experiments is to assess the ability of the proposed approach to correctly recommend services for composition and substitution. The accuracy of such recommendation strongly depends on the accuracy of forecasting long-term QoS. We ran our experiments on a 64-bit Windows 10 environment, in a machine equipped with an intel i7 and 12 GO RAM. We used Keras[1] with Google's Tensorflow[2] as back-end for implementing and training the LSTM model. As it is difficult to get the history of real QoS values, we generated synthetic QoS values (disk storage usage) over different periods of time. Generated values are used to train our QoS prediction models.

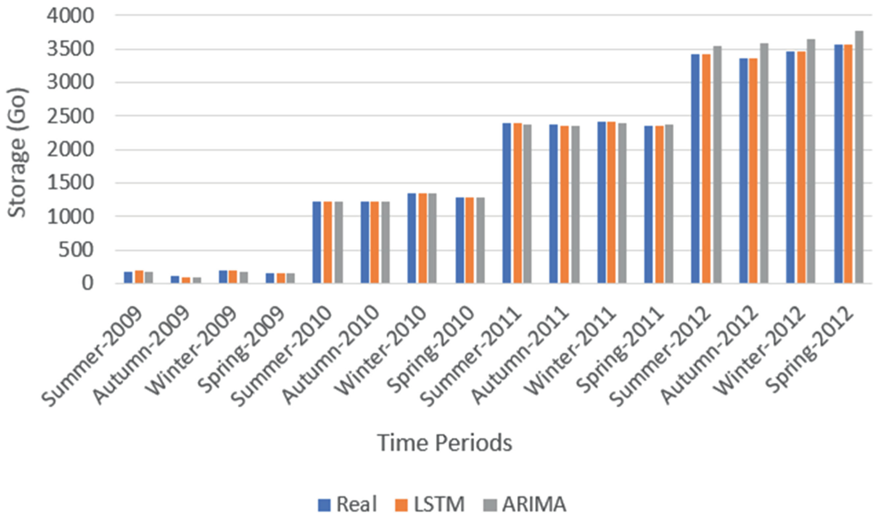

Fig. 2. Comparison of Predicted Storage with LSTMs and ARIMA Models

We compare the accuracy of the proposed prediction with the one that uses ARIMA model [9]. ARIMA (AutoRegressive Integrated Moving Averages) has been successfully used for time series forecasting. Auto regressive means the prediction of $x(t)$ depends on $p = k, p, k \in N$ previous terms. For instance, for $p = 3$, the prediction of $x(t)$ depends on $x(t-1)$,$x(t-2)$, and $x(t-3)$. Moving averages means the prediction depends on the $q = k, q, k \in N$ previous errors.

Figure 2 shows a comparison of the prediction accuracy of our approach (LSTM) and ARIMA. Both models are trained with service storage data from Summer-2009 until Spring-2011 and tested with service storage data from Summer-2012 until Spring-2012. Figure 2 shows that both LSTM and ARIMA achieves comparable accuracy on the training data. However, LSTM outperforms

[1] https://keras.io/.
[2] http://www.tensorflow.com/.

ARIMA on test data. The justification is that LSTM uses many deep hidden layers with non linear transformations among the layers such as *sigmoid* and *tanh* functions. LSTM also saves information longer by using the forget and update gates, hence the bigger the training set, the better LSTM learns and the more accurate is the prediction. On the other hand, ARIMA bases the prediction on p lags which means that the prediction depends on p previous terms.

5 Related Work

In this section, we review the main techniques related to QoS-aware composition of Web services. [9] used the Autoregressive Integrated Moving Average (ARIMA) model to predict future behaviours of the service requests. However, it is not suitable for designing long-term composition as it does not gather stochastic request arrivals. [13] used matrix factorization of a user-service matrix to predict future QoS that can be used for designing compositions. Singular Value Decomposition SVD decomposes the user-service matrix into the product of a user matrix and a service matrix. The reconstructed matrix from the previous product contains the predicted QoS values. Previous approaches rely on instantaneous service QoS values for predicting future QoS that can be used for designing compositions. Our approach uses deep recurrent LSTMs to foresee how QoS values are expected to evolve in time. Such prediction allows providers to better allocate resources to services and developers to better select services for designing long-term compositions. Our approach achieves a more accurate QoS forecasting than the linear methods such as ARIMA [9]. Deep recurrent LSTMs use many hidden layers and non linear transformations between the layers such as *tanh* function. [10] proposed a model for cloud service providers that predicts consumer's service usage behavior (i.e., next requests) and computes the costs of these requests with the goal to maximize cloud service providers incomes. [8] proposes an approach to compose customer requests using the provider long-term qualitative model. Long-term qualitative model is represented as a temporal CP-net. IaaS composition is transformed as a preference maximization optimization problem. [6] defines an approach for long-term QoS-aware cloud service composition. It introduces three meta-heuristic namely Genetic Algorithm, Simulated annealing, and Tabu search to select only services with the best averaged long-term QoS. Our approach relies on deep learning for service QoS prediction. Additionally, we use QoS prediction for both long-term service composition and substitution. LSTMs have been successfully used to solve different prediction problems such as predicting human trajectory in crowded spaces [1]. To the best of our knowledge, this is the first work that uses deep learning for designing long-term QoS aware service composition and substitution techniques.

6 Conclusion

In this paper, we proposed a deep learning approach for service composition using long-term predicted QoS trends. We used deep recurrent Long Short Term Memories (LSTMs) to predict QoS trends over future time periods. The predicted

QoS is used during (i) composition time to ascertain that selected components statisfy developers' QoS requirements in the long run and (ii) substitution to verify that a component and its potential substitute have similar QoS trends. Experiments conducted over synthetic data show that the use of LSTMs for QoS prediction outperforms other techniques such as ARIMA.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 961–971 (2016)
2. Ardagna, D., Casale, G., Ciavotta, M., Pérez, J.F., Wang, W.: Quality-of-service in cloud computing: modeling techniques and their applications. J. Internet Serv. Appl. **5**(1), 11:1–11:17 (2014)
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
4. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benbernou, S.: I, B., Kertész, A., Parkin, M., Carro, M.: A survey on service quality description. ACM Comput. Surv. **46**(1), 1:1–1:58 (2013)
5. Lemos, A.L., Daniel, F., Benatallah, B.: Web service composition: a survey of techniques and tools. ACM Comput. Surv. **48**(3), 33:1–33:41 (2016)
6. Liu, S., Wei, Y., Tang, K., Qin, A.K., Yao, X.: Qos-aware long-term based service composition in cloud computing. In: IEEE Congress on Evolutionary Computation (CEC) 2015, pp. 3362–3369 (2015)
7. Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A.H.H., Elmagarmid, A.K.: Business-to-business interactions: issues and enabling technologies. VLDB J. **12**(1), 59–85 (2003)
8. Mistry, S., Bouguettaya, A., Dong, H., Erradi, A.: Qualitative economic model for long-term iaas composition. In: International Conference on Service-Oriented Computing, pp. 317–332 (2016)
9. Mistry, S., Bouguettaya, A., Dong, H., Qin, A.K.: Predicting dynamic requests behavior in long-term iaas service composition. In: IEEE International Conference on Web Services (ICWS) 2015, pp. 49–56. IEEE (2015)
10. Mistry, S., Bouguettaya, A., Dong, H., Qin, A.: Metaheuristic optimization for long-term iaas service composition. IEEE Trans. Serv. Comput. (2017)
11. Wang, S., Zhu, X., Yang, F.: Efficient qos management for qos-aware web service composition. Int. J. Web Grid Serv. **10**(1), 1–23 (2014)
12. Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. **30**(5), 311–327 (2004)
13. Zhu, J., He, P., Zheng, Z., Lyu, M.R.: Towards online, accurate, and scalable qos prediction for runtime service adaptation. In: IEEE 34th International Conference on Distributed Computing Systems (ICDCS) 2014, pp. 318–327 (2014)