A Context-Aware Approach for Personalised and Adaptive QoS Assessments

Lina Barakat(✉), Adel Taweel, Michael Luck, and Simon Miles

Department of Informatics, King's College London, London, UK
{lina.barakat,adel.taweel,michael.luck,simon.miles}@kcl.ac.uk

Abstract. Given the importance of QoS (quality of service) properties for distinguishing between functionally-equivalent services and accommodating different user expectations, a number of QoS estimation approaches have been proposed, utilising the observation history available on a service. Although the context underlying such previous observations (and corresponding to both user and service related factors) could provide an important source of information for the QoS estimation process, it has only been utilised to a limited extent by existing approaches. In response, we propose a context-aware quality learning model, realised via a learning-enabled service agent, exploiting the contextual characteristics of the domain in order to provide more personalised, accurate and relevant quality estimations for the situation at hand. The experiments conducted demonstrate the effectiveness of the proposed approach.

Keywords: Context awareness · Change detection · Personalisation · Quality value learning

1 Introduction

Services advertised by different providers can overlap in their functional capabilities, but offer varying quality of service (QoS) levels. Such QoS properties, thus, play an essential role in differentiating between functionally equivalent services and accommodating different user needs. However, the subjective nature of some properties and the dynamic and unreliable nature of service environments may result in cases where the quality values available from the service provider are either uninstantiated or untrustworthy. Consequently, a number of efforts focus on learning more accurate estimation of service quality values, based on the data available regarding the service's past performance (e.g. [5,7,8]). Most such learning approaches, however, rely on data recency to account for potential changes in the service's behaviour. That is, newer service observations are favoured, while older ones are eventually forgotten, without consideration of the service's circumstances, thus neglecting important evidence for detecting a change occurrence. Moreover, the observations are usually assumed to be objective, and thus the predictions produced do not account for a user's particular situation. We argue that acccounting for the circumstances under which the observations were collected (in relation to both the user and the service) is essential to ensure that only relevant data is captured in the learning process, as illustrated below.

User Circumstances. Consider a scenario where a user wants to order a meal for dinner, and therefore contacts a food-specialised broker. The broker has access to information about a pool of meal delivery services that are offered by various food providers. Let's assume that the user suffers from chewing and swallowing problems, and therefore requires the meal to be of tender texture. Given a candidate meal option, the broker thus needs to assess its corresponding texture from past available ratings to determine its suitability for the user. Since the perception of food texture could be affected by the presence of chewing and swallowing difficulties, the ratings of users sharing similar dysphagia conditions with the current user should have the highest impact on texture assessment at hand, while the contribution of those with no difficulties should be minimal.

Service Circumstances. Consider a similar food ordering scenario where a user is interested in a meal that is highly rated in terms of taste. Again, the broker here needs to assess the taste property of each candidate meal option. Assume one such option is service s, with a good rating history up to time step t_k, after which the service exhibits a change in recipe, occurring, for instance, due to a change in the head chef, or the temporary unavailability of some ingredients (e.g. some ingredients might not be available at winter time). Such a change could affect many aspects of the meal, including taste, making such previous user observations under the old recipe less (or no longer) relevant under the new one. Now, if the service switches again to the old recipe, window $[t_1, t_k]$ of the historical observations on taste available for service s becomes relevant again, and is a useful source of information for assessment of taste for this service.

Given this, we propose enriching service observations with contextual information, and exploiting such information during QoS learning to capture the most relevant data for the situation at hand, thus achieving more personalised and adaptive quality predictions. By context, we refer to any conditions and circumstances that may affect the perception of a quality value by a service user, either related to the user itself (user context) or related to the service (service context). The context model is presented in Sect. 2. Sections 3 and 4 present our context-aware QoS learning model and the experimental results, respectively. Related work and conclusion are discussed in Sects. 5 and 6.

2 Context Model

The quality characteristics of a service may be dependent on the user situation (user context) under which the service provision happens (e.g. user's location), as well as on service-related circumstances (service context), which could change over time either periodically (e.g. a change in a food service's recipe with season) or non-periodically (e.g. a rare event such as a sudden server crash).

Formally, knowledge of context information relevant for a service provision is a tuple $(Q, C^u, C^s, ctx_u, ctx_s, dom)$, as follows. Q is the set of quality of service attributes of the service, either generic such as price and response time, or domain-dependent such as taste of a food service. C^u and C^s are the sets of attributes characterising a user's context and the service's context, respectively, expected to affect the quality values delivered by the service (and are shared among similar service types). ctx_u is a quality attribute's user context function, mapping quality attribute $q \in Q$ to the user-related context attributes $ctx_u(q) \subset C^u$ that may have an impact on the perception of q by the user, e.g., $ctx_u(\text{texture}) = \{\text{chewing and swallowing condition}\}$, and $ctx_u(\text{price}) = \emptyset$. ctx_s is a quality attribute's service context function, mapping quality attribute $q \in Q$ to the service-related context attributes $ctx_s(q) \subset C^s$ that may have an impact on the behaviour of the service so that the same user may observe different values of q under different values of these attributes, e.g., $ctx_s(\text{taste}) = \{\text{food recipe}\}$. Finally, dom is an attribute domain function, mapping an attribute a (quality or contextual) to its corresponding set of possible values. In this paper, we assume that $dom(a)$ corresponds to a discrete domain (for continuous attributes, this is obtained via applying an appropriate discretisation algorithm).

3 Context-Aware QoS Learning

In our approach, a service's QoS characteristics are assessed via a learning-enabled agent associated with the service. This agent (which, for example, could be acting on behalf of the service provider or the broker with which the service is registered) exploits a contextually-enriched history of past interactions with the service in order to expose personalised and dynamism-aware QoS information to clients. The modelling and learning details of such an agent are presented next.

3.1 Service Observation

For each quality attribute $q \in Q$, the agent receives a stream of service observations, each reporting the outcome encountered for q in a previous interaction with the service, along with the contextual circumstances surrounding this interaction. Formally, a service observation is denoted as $(v_q, \boldsymbol{v_u}, \boldsymbol{v_s})$, where: $v_q \in dom(q)$ is the value observed for q; $\boldsymbol{v_u} = (v_u^1, ..., v_u^m)$ are the respective values of user-side contextual attributes $(c_u^1, ..., c_u^m) \in ctx_u(q)^m$; and $\boldsymbol{v_s} = (v_s^1, ..., v_s^k)$ are the respective values of service-side contextual attributes $(c_s^1, ..., c_s^k) \in ctx_s(q)^k$.

3.2 Agent Configuration and Learning Model

The main idea behind our approach is that, for a particular quality attribute, the agent maintains a set of learned *value models*, each corresponding to a different behaviour of the service (as a result of changes in service-side circumstances). When previously-encountered service circumstances reoccur, older observations of the service collected under such circumstances become relevant again, and the agent can reuse the respective historical value model (learned from these observations) to make future quality value predictions. Such reuse of a previously learned value model facilitates a faster adaptation to a behavioural change of the service (as opposed to re-learning the behaviour from new interactions), and consequently improves the accuracy of quality predictions.

Formally, the configuration of a service agent at a particular time step is a tuple $(\Omega, \mathit{active})$, where: Ω is the model library of the agent, containing the set of learned value models for quality attributes; and function *active* maps each quality attribute $q \in Q$ to its currently active value model $\mathit{active}(q) \in \Omega$ (i.e. the model utilised to predict the attribute's value for the next discovery attempt by a consumer). Each model $\omega \in \Omega$ is of the form $q : \psi \leftarrow M$. Here, $q \in Q$ is the quality attribute the value of which the model is trying to predict. Precondition ψ identifies the service-side contextual circumstances under which the model is valid (it is a logical formula in disjunctive normal form (DNF) restricting the values of contextual attributes $c_s \in \mathit{ctx}_s(q)$). Finally, body M is the actual prediction model for quality attribute q under condition ψ. It corresponds to the underlying function *qval* between the values of user-side contextual attributes affecting q and the corresponding value of q, i.e. $\mathit{qval}(q, \boldsymbol{v_u}) \in \mathit{dom}(q)$ is the value predicted for quality attribute q given user's contextual values $\boldsymbol{v_u}$.

The configuration of the agent evolves over time as the agent's learning progresses. In particular, given the current configuration $(\Omega, \mathit{active})$, and a new service observation $(v_q, \boldsymbol{v_u}, \boldsymbol{v_s})$, value vector $\boldsymbol{v_s}$ is compared against the contextual precondition ψ of the currently active model $\mathit{active}(q)$, and two cases are distinguished. Case 1. $\boldsymbol{v_s}$ is subsumed by ψ, in which case no behavioural drift is assumed, and observation $(v_q, \boldsymbol{v_u})$ is simply used to update body M of the currently active model $\mathit{active}(q)$ to increase its accuracy with more incoming data. Case 2. $\boldsymbol{v_s}$ is not subsumed by ψ, in which case a behavioural drift is suspected and further two sub-cases are distinguished. Case 2.1. $\boldsymbol{v_s}$ is subsumed by the contextual precondition ψ' of another existing model $\omega' \in \Omega$ of attribute q $(\omega' \neq \mathit{active}(q))$. Here, a recurring behaviour (i.e. a behaviour learned previously) is assumed, and the respective model ω' becomes the current active model for attribute q, with its body M' being updated with observation $(v_q, \boldsymbol{v_u})$. Case 2.2. $\boldsymbol{v_s}$ is not subsumed by any contextual precondition of any previously learned model for quality attribute q. In this case, the agent suspects a new service behaviour, and therefore set up a new model ω^n for attribute q, which is added to the model library. The contextual precondition ψ^n of this model is the conjunction of values $\boldsymbol{v_s}$, while its body M^n is built incrementally from the new incoming observations starting from the current observation $(v_q, \boldsymbol{v_u})$.

After the new model ω^n is stabilised (i.e. after *stability* incoming observations under condition ψ^n), it is compared against the other existing models in the agent's library to verify whether it is actually reflecting a new service behaviour for attribute q (we utilise the *conceptual equivalence measure* proposed by Yang et al. [1] for such comparison). If no similar model is found, the new behaviour is confirmed and model ω^n becomes the currently active model for attribute q. Otherwise (i.e. a similar model ω^{sim} exists in the library), model ω^n is discarded (i.e. removed from the library), while model ω^{sim} is regarded as the currently active model for q, with its contextual precondition ψ^{sim} being generalised to subsume condition ψ^{n}. Note that, if a service context different to $\boldsymbol{v_s}$ is encountered prior to stabilising model ω^{n}, the observations encountered under condition ψ^{n} are considered as noise and ω^{n} is simply discarded.

4 Experiments and Results

We evaluate the performance of the proposed approach in terms of producing accurate quality value predictions in dynamic and user-dependent settings[1]. We show the results (averaged over 100 runs) from the perspective of one service and one quality attribute q. An experiment run consists of a number of learning episodes of the service agent, each involving three steps: (1) *observing* value v_q for quality attribute q delivered by the service under user's context $\boldsymbol{v_u}$ and service's context $\boldsymbol{v_s}$; (2) *adjusting* the current configuration utilising this new observation $(v_q, \boldsymbol{v_u}, \boldsymbol{v_s})$; and (3) *predicting* the expected quality value for the next user using the adjusted configuration. Further details are presented next.

4.1 Value Model Implementation

To implement the body M of each model $\omega \in \Omega$, we use the Naive Bayesian classifier [2]. In particular, given a quality attribute q and a corresponding observed user's context sample $\boldsymbol{v_u}$, the value v_q predicted for q is the one maximising the posterior probability $p(v_q|\boldsymbol{v_u})$, given as $p(v_q|\boldsymbol{v_u}) = \frac{p(v_q)\times p(\boldsymbol{v_u}|v_q)}{p(\boldsymbol{v_u})}$. Here: $p(v_q)$ is the prior probability of value v_q; $p(\boldsymbol{v_u})$ is the prior probability of sample $\boldsymbol{v_u}$ (this is the same for all the values of q and thus could be omitted); and $p(\boldsymbol{v_u}|v_q)$ is the posterior probability of sample $\boldsymbol{v_u}$ conditioned on value v_q. To simplify the computation cost of $p(\boldsymbol{v_u}|v_q)$, independence is usually assumed among the attributes of the sample, leading to: $p(\boldsymbol{v_u}, v_q) = \prod_{i=1}^{m} p(v_u^i|v_q)$. The estimation of probabilities $p(v_q)$ and $p(v_u^i|v_q)$ can be easily achieved via maintaining corresponding value counts, and thus the incremental learning function $learn_{q,u}$, corresponds to the update of these counts after each new service observation.

4.2 Dataset

We utilise a synthetic dataset inspired by STAGGER concepts [2], but is adapted to suit our problem. We assume: five possible outcomes for quality attribute q, $dom(q) = \{v_q^1, v_q^2, v_q^3, v_q^4, v_q^5\}$; three user-side context attributes, c_u^1, c_u^2, and c_u^3, affecting q, each with three possible values, $dom(c_u^1) = \{v_u^{1,1}, v_u^{1,2}, v_u^{1,3}\}$, $dom(c_u^2) = \{v_u^{2,1}, v_u^{2,2}, v_u^{2,3}\}$, and $dom(c_u^3) = \{v_u^{3,1}, v_u^{3,2}, v_u^{3,3}\}$; one service-side context attribute, c_s, with three possible values, $dom(c_s) = \{v_s^1, v_s^2, v_s^3\}$; and three different service behaviours regarding attribute q, *behaviour 1* (associated

[1] Source code and data for the results presented in this paper are freely available from http://jaspr.org/source-code.

with v_s^1) where the actual value of q is v_q^1 (under $v_u^{1,1} \wedge v_u^{2,1}$) and is v_q^2 (otherwise), *behaviour 2* (associated with v_s^2) where the actual value of q is v_q^1 (under $v_u^{2,2} \vee v_u^{3,2}$) and is v_q^3 (otherwise), and *behaviour 3* (associated with v_s^3) where the actual value of q is v_q^4 (under $v_u^{1,2} \vee v_u^{1,3}$) and is v_q^5 (otherwise). The service switches from one behaviour to another at particular points, with such drifts being associated with changes in the value of service-side context attribute c_s.

4.3 Evaluation Strategies and Measure

We refer to the following quality value learning strategies. Strategy MML, our proposed multi-model learning approach. Strategy SSL, a simple summary-based learning approach, which predicts the quality value v_q with the highest prior probability, $p(v_q)$, based on all the observations so far and ignoring the user's context. Finally, strategy SWL_w, a sliding window based learning approach, a well known way in the literature of adapting to potential changes in incoming data [3]. It utilises the Naive Bayesian classifier presented in Sect. 4.1 as its main model, but maintains a fixed window of the latest w observations, based upon which the model is updated at each time step (this strategy accounts for the user's context, but ignores the service's context). By SWL_all, we refer to accounting for *all* the data observed so far. The performance of each strategy is evaluated by assessing its prediction accuracy at each time step, calculated as the success rate (i.e. $\frac{\text{number of successful predictions}}{\text{total number of predictions}}$) over the last 20 observations.

4.4 Results

To study the importance of user context awareness, we first assume a static service behaviour regarding attribute q (e.g. *behaviour 2*), and compare our learning strategy, MML, against the simple summary one, SSL. To simulate situations of imperfect user's contextual knowledge, attribute q is subjected to different levels of noise η%. The results in Fig. 1(a) demonstrate that MML achieves an accuracy of over 80 % (at 0 % noise) after only 30 cycles, as opposed to SSL where the accuracy fluctuates around 50 % for the entire run. It is also evident that even with high noise levels (e.g. 30 % noise), MML still outperforms SSL, indicating the importance of contextual evidence even if imperfect.

To study adaptivity in dynamic environments, we now assume that the service follows the behaviour sequence 1-2-3-1-2-3, with each behaviour being fixed for 300 episodes. Figure 1(b) compares the adaptation strategies in the case of encountered *new behaviour* (i.e. changes during the first 900 episodes), with *stability* being set to 15. MML outperforms the other strategies, increasing the accuracy to over 90 % after just 30 observations from the change point. SWL_all, however, suffers from poor performance, especially after a change, where the learned model mostly reflects irrelevant observations. In fixed windowing strategies, increasing the window size results in a slower reactivity to a change since older irrelevant observations take longer to be forgotten, while smaller windows achieve faster adaptation but affect the prediction accuracy due to depending on insufficient number of observations. In the case of encountered

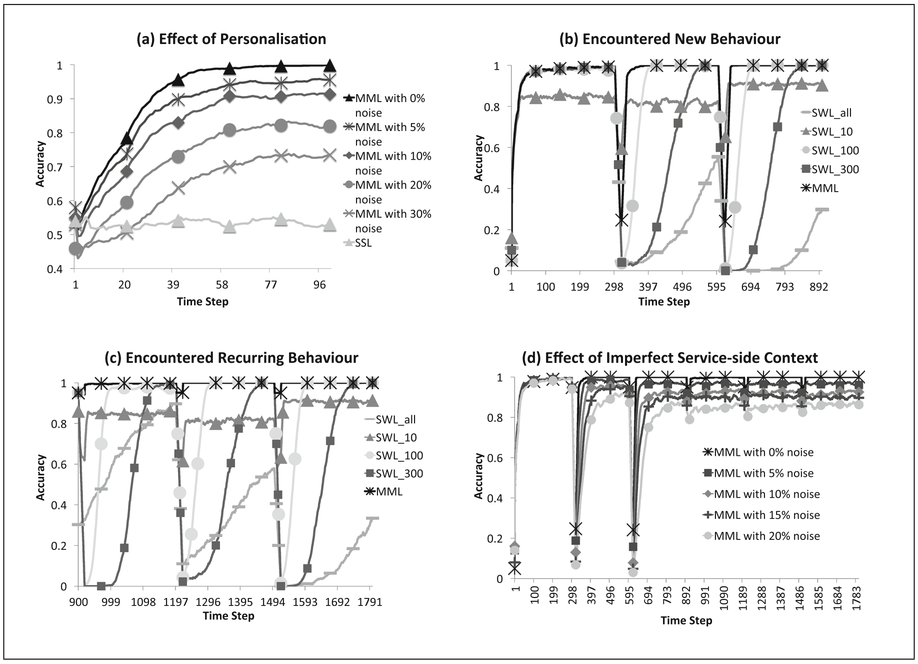

Fig. 1. Evaluation results

recurring behaviour (i.e. changes in the second 900 episodes), and unlike the other strategies, MML always maintains high accuracy (see Fig. 1(c)), eliminating the period of performance degradation after a change due to reusing an existing stable model. The effect of imperfect (e.g. incomplete) service-side context knowledge on MML is studied in Fig. 1(d), where context attribute c_s is subjected to various levels of noise $\eta\%$, and the results indicate robustness to noise.

5 Related Work

The QoS properties of services are important criteria upon which services are discovered, selected, and composed [4]. Accurate estimation of such properties has thus received much attention. Aschoff et al. [5] model the response time of a service as a random variable, with the exponentially weighted moving average being utilised for estimating the expected value of this variable at a particular time step according to historical data. Similarly, time series modelling based on ARIMA (AutoRegressive Integrated Moving Average) has been proposed by Amin et al. [6] for the purpose of QoS forecasting. Barakat et al. [7] provide probabilistic, multi-valued quality estimations for services via applying an online learning algorithm inspired by Policy Hill-Climbing based on past user ratings. Trust and reputation mechanisms have also been considered for the purpose of accurate quality predictions [8], where an assessment of a QoS dimension's trustworthiness (e.g. a time-weighted average of the past service ratings) is undertaken prior to an interaction. In contrast to our approach, all such efforts rely on favouring recent observations to handle changes in the service's behaviour, without accounting for contextual clues, thus neglecting important evidence.

To facilitate personalised QoS information for users, a number of approaches utilise collaborative filtering for quality value prediction [9]. Although such approaches capture the user's context implicitly, they usually suffer from the data sparsity problem, unlike our approach, which explicitly exploits available user's contextual knowledge, which allows deriving *personalised* quality assessments for the user even in the absence of previous interactions with this user. Finally, like us, Lin et al. [10] explicitly incorporate the user's contextual attributes as input for the quality value prediction process. Yet, they do not account for changes in the service's behaviour associated with service-side context.

6 Conclusion

The paper presented a context-aware QoS learning approach for personalised and adaptive quality estimations. The learning is conducted via a service agent, which maintains a pool of quality prediction models; each characterising a particular service behaviour and providing personalised value predictions for users. Experimental results show that the approach achieves high accuracy and a faster change adaptation when compared to commonly adopted time-based learning.

Acknowledgments. This work was part funded by the UK Engineering and Physical Sciences Research Council as part of the Justified Assessments of Service Provider Reputation project, ref. EP/M012654/1 and EP/M012662/1.

References

1. Yang, Y., Wu, X., Zhu, X.: Mining in anticipation for concept change: proactive-reactive prediction in data streams. Data Min. Knowl. Discov. **13**, 261–289 (2006)
2. Widmer, G.: Tracking context changes through meta-learning. Mach. Learn. **27**, 259–286 (1997)
3. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**, 1–37 (2014)
4. Barakat, L., Miles, S., Poernomo, I., Luck, M.: Efficient multi-granularity service composition. In: IEEE International Conference on Web Services, pp. 227–234 (2011)
5. Aschoff, R., Zisman, A.: QoS-driven proactive adaptation of service composition. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) Service Oriented Computing. LNCS, vol. 7084, pp. 421–435. Springer, Heidelberg (2011)
6. Amin, A., Colman, A., Grunske, L.: An approach to forecasting QoS attributes of web services based on ARIMA and GARCH models. In: IEEE International Conference on Web Services, pp. 74–81 (2012)

7. Barakat, L., Mahmoud, S., Miles, S., Taweel, A., Luck, M.: An agent-based service marketplace for dynamic and unreliable settings. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 169–183. Springer, Heidelberg (2014)
8. Maximilien, E. M., Singh, M. P.: Agent-based Trust Model Involving Multiple Qualities. In: 4th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 519–526 (2005)
9. Zheng, Z., Ma, H., Lyu, M.R., King, I.: QoS-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**, 140–152 (2011)
10. Lin, D., Shi, C., Ishida, T.: Dynamic service selection based on context-aware QoS. In: IEEE International Conference on Services Computing, pp. 641–648 (2012)