Quantitative Evaluation of QoS Prediction in IoT

Gary White, Andrei Palade, Christian Cabrera, Siobhán Clarke
School of Computer Science and Statistics,
Trinity College Dublin, Dublin, Ireland
{whiteg5, paladea, cabrerac, siobhan.clarke}@scss.tcd.ie

Abstract—Internet of Things (IoT) applications, are typically built from services provided by heterogeneous devices, which are potentially resource constrained and/or mobile. These services and applications are widespread and a key research question is how to predict user side quality of service (QoS), to ensure optimal selection and composition of services. Invoking all available IoT services for evaluation is impractical due to the exponential growth in these services. To assess the current state of the art related to this research question we conduct a quantitative evaluation of QoS prediction approaches, particularly those that use matrix factorisation (MF) for collaborative QoS prediction. These approaches derive from the collaborative filtering model used in recommender systems, which avoids the problems of many other QoS prediction approaches of requiring additional invoking of available IoT components on behalf of the user. We conduct comprehensive experiments based on a real-world large-scale QoS dataset as well as a transformation of this dataset to more closely estimate IoT services, to show the prediction accuracy of these approaches. We also give a demonstration of how they can be used in a small scale example.

I. INTRODUCTION

The emergence of a new generation of cheaper and smaller wireless devices with a number of communication protocols has led to the formation of the IoT. The number of these devices is predicted to grow at an exponential rate, with the latest forecasts predicting that there will be between 26 and 50 billion connected devices by 2020 [1], [2]. The huge number of devices in the IoT will lead to a wide variety of services, which can provide information from a number of sources such as home applications, surveillance cameras, monitoring sensors, and actuators [3]. This leads to applications in many different domains, such as home automation, industrial automation, medical aids, traffic management and many others [4].

These applications have a range of QoS requirements, which can be typically categorised as best effort (no QoS), differentiated services (soft QoS) and guaranteed services (hard QoS). As these devices are usually resource constrained [5] there is a high probability that the QoS provided will fail or degrade during service execution. These devices are capable of providing multiple services, which means that there will be a large number of functionally similar services available to users. A key requirement is to be able to select the best service based on the non-functional requirements such as QoS, either at design time, or runtime using runtime service adaptation, if the currently executing service begins to degrade. This is especially important for safety critical services such as those in healthcare, where a service failing can have a serious impact.

Switching to a service with better QoS is not a simple task as the QoS of a service takes into account a number of different factors such as response time and throughput, which may be influenced by user preference. There are also a number of QoS factors, which are difficult to quantify such as the security and functional stability of a service. Quantifiable QoS factors, such as response time and throughput, are not stable and vary by time and location of invocation [6]. A number of approaches have been proposed to deal with these challenges and to make predictions of QoS values based on time-series analysis [7], [8], [9]. These approaches make predictions about the future QoS they will receive based on the current values being monitored, but they cannot make predictions on the QoS of services that they have not invoked, unless they start executing them. This is a problem in the IoT as there will be a huge number of candidate services, which could be invoked and when a currently executing service begins to degrade, we need to be able to find an alternative functionally equivalent service with the requested QoS, without having to invoke all possible services.

An alternative approach to predicting QoS is inspired from recommender systems, where the QoS information of similar users is used to make predictions about the QoS from possible services, by collaborative filtering. These approaches use MF to allow the user to receive QoS predictions of services that they have not invoked, based on the QoS values from similar users. Using the QoS from other similar users gives more information about candidate services, which can be chosen either at design time or during runtime service execution. This also has the additional benefit that we can gather the additional QoS information without harming the performance of the infrastructure with needless invocations of services to retrieve the QoS. This is a key challenge in SOAs [10], especially in IoT where devices can be resource constrained and/or mobile.

MF algorithms have not been used in the IoT and in this paper we conduct a comprehensive evaluation of existing MF methods on a dataset of QoS information from real world services. We also conduct experiments with a transformation of this dataset to show how the algorithms perform with data, which more closely resembles what would be found in the IoT, with more service failures and longer delay times. The remainder of the paper is organised as follows: Section II describes the collaborative framework in IoT, Section III describes the experiments that were conducted, Section IV describes the results of the experiments and Section V outlines the conclusions and future work.
QoS information from a number of different users is needed to use collaborative filtering. Figure 1 shows our evaluation middleware architecture, including the QoS monitor used to collect QoS information from a number of different user requests. The QoS prediction component is implemented using a number of different MF algorithms. Figure 2 illustrates where the middleware is implemented in gateways that execute services from heterogeneous service providers. This figure shows a typical IoT scenario, with services provided from different service types including web services, wireless sensor networks (WSN), and autonomous service providers (ASP). The services have a range of different QoS properties including response-time, throughput and availability. The communication links for invoking these services are diverse, which will affect the personal QoS experience of the users, so are monitored using the monitoring engine. These monitored results are used by the prediction engine to provide personalised QoS prediction to service consumers, which is the focus of this paper.

Within the middleware there are several components which are used to handle requests from service consumers. The request handler exposes a request interface to the service consumers, which then subscribe to response messages that are published when the execution has finished. The service registration engine collects the service description information by subscribing to WSN providers, which publish their service description and expose a service to register web services. This is stored in a service registry, which the service discovery engine searches and publishes messages to the service composition and execution engine if the services are available. The QoS monitor provides personalised QoS prediction of the selected services for the user and recommends the best service to the service execution engine. The service execution engine then invokes the components and publishes the results to the request handler when the execution finishes.

The QoS monitor contains two sub-components: A Monitoring Engine, which is responsible for collecting a number of QoS factors from users such as the response time, cost and throughput and the Prediction Engine which provides the personalised QoS prediction values for different component users. In this paper, we focus on the design of the prediction engine and quantitatively evaluate a number of algorithms based on their prediction accuracy.

Even in a small scale scenario such as the demonstration shown in Figure 2, traditional time-series approaches can cause a number of additional service requests for each user which can flood the network. The collaborative framework allows for users to share local QoS usage experiences and combine this information to get a global view of the QoS of the services. We can then use algorithms such as MF to make predictions of QoS values for new users based on local information from similar users and global information from the service providers. Each user keeps a record of their QoS experiences in the service registry. If a service consumer would like to get personalised QoS information, they can allow their QoS records to be used for QoS prediction. Based on the collected QoS information, the Prediction Engine can then make personalised QoS prediction for each individual user and forward the recommended services to the service execution engine.

### III. EXPERIMENTAL SETUP

#### A. Small Scale Demonstration

To provide QoS values on \( m \) IoT services for \( n \) users, you need to invoke at least \( n \times m \) services, this is almost impossible
in an IoT environment where we expect a large number of services and users. Without this QoS information, the service execution engine cannot select the optimal components based on their QoS and must make a choice based on whatever services are available. This leads to choosing potentially non-optimal services which can cause service degradation at runtime and service execution errors.

To address these challenges, we have conducted a quantitative evaluation of matrix factorisation approaches, for personalised QoS prediction in IoT on an open source dataset [6] as well as a small scale demonstration based on Figure 2. The approaches use ideas from recommender systems where users that share similar characteristics (e.g., location, response time, etc.) will receive similar QoS when executing the same service. The QoS value of IoT service \( s \) observed by user \( u \) can be predicted by exploring the QoS experiences from a user similar to \( u \). A user is similar to \( u \) if they share similar characteristics, which can be extracted from their QoS experiences on different components by performing non-negative matrix factorisation (NMF). By sharing local QoS experience among users these approaches can predict the QoS value of a range of IoT services which include ASP, web services and WSN even if the user \( u \) has never invoked the service \( s \) before.

We demonstrate an example based on the implementation in Figure 2, where we have a number of different service providers, who are able to provide functionally equivalent services from heterogeneous devices. We model this as a bipartite graph \( G = (U \cup S, E) \), such that each edge in \( E \) connects a vertex in \( U \) to \( S \). Let \( U = \{u_1, u_2, ..., u_N\} \) be the set of component users and \( S = \{ASP_1, ASP_2, ..., WSN_N\} \) denote the set of IoT services and \( E \) (solid lines) represent the set of invocations between \( U \) and \( S \). Given a pair \((i, j)\), \( u_i \in U \) and \( c_j \in S \), edge \( e_{ij} \) corresponds to the QoS value of that invocation. Given the set \( E \) the task is to predict the weight of potential invocations (broken lines).

We visualise the process of matrix factorisation for the demonstration in Figure 3b, in which each table entry shows an observed weight in Figure 3a. The problem that matrix factorisation solves is how to predict the missing values in the user service matrix based on the existing entries. These predicted entries can then provide users with personalised QoS information which can be used for service selection in service composition at both design and runtime.

One of the problems that emerge due to the large amount of potential invocations in the IoT is that there are a number of functionally similar services for users to choose, which reduces the possibility of having commonly invoked services. Since the similarity of users is calculated by comparing common services, the problem of having few common services makes it difficult to calculate the similarity between users. By using the latent factor model [11] a number of algorithms have been able to first factorise the sparse user-component matrix and then use \( V^T H \) to approximate the original matrix, where the low-dimensional matrix \( V \) denotes the user latent feature space and the low-dimensional matrix \( H \) represents the low-dimensional item latent feature space. The latent feature space represents the underlying structure in the data, computed from the observed features using matrix factorisation. The rows in the two matrices represent different features with each column in \( V \) representing a user and each column in \( H \) denoting a service. As the matrices \( V \) and \( H \) are dense it is then possible to use a neighbourhood-based collaborative method, as shown in Figure 3c.

### B. Dataset Description

Invoking thousands of IoT services in the wild is difficult because some of the services may have limited range and may not be available on the Internet. To evaluate the prediction quality of these approaches, we use the dataset released by Zheng et al. [6], which consists of a matrix of the response time and throughput of 339 users by 5,825 web services. We also conduct experiments with a transformation of this dataset, which more closely resembles services in IoT, where there is a greater distribution in response time and throughput and there are more device failures. We transform the response time and throughput matrix by adding random values from a poisson distribution, which is standard practice for modelling network traffic in cellular networks and the IoT [12].

### C. Metrics

To evaluate the prediction accuracy of the proposed algorithms, we use two standard error metrics, the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE). The metric MAE is defined as:

\[
MAE = \frac{1}{N} \sum_{i,j} |w_{i,j} - \hat{w}_{i,j}|	ag{1}
\]

and RMSE is defined as:

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i,j} (w_{i,j} - \hat{w}_{i,j})^2}	ag{2}
\]

where \( w_{i,j} \) is the QoS value of service \( c_j \) observed by user \( u_i \), \( \hat{w}_{i,j} \) denotes the predicted QoS value of the service \( c_j \) by user \( u_i \). \( N \) is the number of predicted QoS values. \( MAE \) gives equal weight to individual differences, \( RMSE \) on the other hand gives high weights to large errors due to the squaring term.

### D. Evaluated Algorithms

The MF algorithms used to make the QoS predictions of the web services and the dataset which we have transformed, are as follows:

1. **CloudPred**: A neighbourhood-based approach, enhanced by feature modelling on both users and components. [13].
2. **EMF** (Extended Matrix Factorisation): A framework with relational regularization to make missing QoS value predictions [14].
3. **LN_LFM** (Latent Factor Models): A Latent Factors Model, which can be utilised to predict the unknown QoS values [15].
4) NMF (Non-negative Matrix Factorisation): A non-negative matrix factorisation approach on user-item matrix for missing value prediction [16].

5) PMF: A personalised neighbourhood matrix approach [17].

These algorithms represent a number of alternative approaches to MF, which we consider to be the current state of the art, based on results in their respective papers.

IV. RESULTS

To evaluate the performance of the different approaches, we remove random entries from the matrices and compare the values which were predicted to the original ones. The prediction accuracy is calculated using Equation (1) and (2). To evaluate the impact of matrix density we use matrices with missing values in different densities. For example, 10% means that we randomly remove 90% entries from the original matrix and use the remaining 10% to predict the original values. To test the impact of the dataset we conduct experiments on a transformation of the original dataset as described in Section III-B, which more closely models what would be expected in the IoT with longer tails in the distribution. Finally, we conduct experiments on the dimensionality of the algorithms, which determines the number of latent features used to characterise the user and the IoT services. This parameter allows for the tuning of the individual algorithms to reduce error in certain conditions such as a low matrix density.

A. Impact of Matrix Density

Table I and Figure 5 show the impact of matrix density on the prediction accuracy. To estimate the impact of the matrix density, we randomly remove some entries from the matrices and compare the predicted values to the original ones. We test a number of different matrix densities, we change the matrix density from 10% to 90% with a step value of 10% and repeat each step sizes 20 times. In Figure 5 we draw the standard deviation in the region around the average error for the 20 trials. The parameter settings for each of the algorithms are set to the default parameters mentioned in their respective papers.

Figure 5a and 5b show the experimental results of response time, while Figure 5c and Figure 5d show the experimental results conducted on the throughput dataset. We can see from the figures that matrix density has a large impact on prediction accuracy, with an increase in density improving the prediction accuracy of all the algorithms. In an IoT environment, we would expect the matrix density to be quite low due to the large number of potential services, so it is important to identify how these algorithms perform with low matrix density.

Table I shows the detailed results of the experiments, where the best performing result for each density is in bold. For the response-time dataset the CloudPred algorithm has performed the best for all matrix densities in both the MAE and RMSE. Figure 5a and 5b visualise the response time results from the table and show how each of the algorithms follow a similar slope as the matrix density increased.

The throughput dataset shows more varied results with the best algorithm switching from NMF, EMF and LN LFM for different matrix densities as can be seen in Table I. We can also see that the errors are much larger for the throughput dataset, due to the scale of data, which varies more than the response time as can be seen in Figure 4. One solution may be to use an ensemble of the existing algorithms to reduce error. The overall results from this experiment show how much of an influence matrix density has on these algorithms, and the amount of error to be expected when there is low matrix density, which may be the case in IoT.

B. Impact of Dataset

As mentioned in Section III-B we transform the original dataset using random values drawn from a poisson distribution to more closely resemble QoS values which would be expected in the IoT. We use the CloudPred algorithm to demonstrate the effect of the change in dataset and the matrix density from
10% to 90% with a step value of 10%. The parameters for this experiment are Top-K=10, dimensionality=20 and λ=0.5, we repeat the experiment 20 times and draw the standard deviation in the region around the average error.

Figure 6 shows the impact that the change in dataset has had on the results, we can see that for the response time dataset the change has increased the error considerably with over twice the amount of error for the transformation by lambda=1, 2 in the MAE as can be seen in Figure 6a and the RMSE in Figure 6b. For the throughput dataset, the change has had much less of an impact and the error values remain consistent in the MAE as can be seen in Figure 6a and the RMSE in Figure 6c. The impact of the change in dataset is influenced by the original data, as the scale of the response times is much smaller, 0-20 seconds compared to throughput, which is 1-1000kbps as can be seen in Figure 4. This suggests that the QoS factors which the algorithms are used on can have an impact on accuracy of the results. This is something to take into consideration when using these algorithms to model a number of factors, which have a large range of values such as throughput as they may not be able to accurately predict within a close bounds.

C. Impact of Dimensionality

The dimensionality parameter is used to determine the number of latent features, which characterise the users and services. We study the impact of this parameter by varying the value of dimensionality from 10 to 50 with a step size of 10 on the CloudPred algorithm. Other parameter settings remain constant and are set to Top-K=10 and λ=0.5 and the matrix density is set to 10%. We repeat the experiment 20 times and draw the standard deviation in the region around the average error.

Figure 7 shows the results of this experiment and the effect that the dimensionality parameter has on the results. We can see that a low value for matrix dimensionality can lead to underfitting and an increase in prediction error for the algorithm. As the dimensionality increases the algorithm adds more latent features to the model which can reduce the accuracy by overfitting to the data. In Figure 7, we see how the error seems to stay constant even as more latent features are added to the model.

V. CONCLUSION AND FUTURE WORK

We have evaluated a number of matrix factorisation approaches and have shown how these approaches can be used to make QoS predictions by exploring the past usage from both the user and other similar users both for web services and a transformed dataset to more closely resemble IoT data. The results illustrated in Figure 5 and 6 show the impact that the density and dataset have on the results. The matrix density has an obvious effect on the prediction accuracy as the error reduces for all of the algorithms as the matrix becomes more dense. The transformation of the dataset to more closely resemble IoT QoS has led to an increase in the prediction error most notably in response time, where the error rate more than doubled. We also show the impact of internal model parameters such as the dimensionality of the algorithm in Figure 7, which shows that the model parameters may have to be tuned as they can have a large impact on the prediction accuracy of the model.

The results have shown that these approaches have increased error for prediction IoT services. However, they have a number of additional benefits which would make them extremely useful in the IoT. They require no additional invocation, which reduces the load on the network. This is a key requirement for resources constrained devices, which are used in the IoT. They also allow the user to utilise both local usage information.
from similar users and global information from the service providers. The IoT is highly dynamic in nature and the QoS performance of services can vary dramatically with time and service providers moving in and out of the environment, causing a service to suddenly not be available either due to mobility or running out of battery. This is in combination with the usual QoS performance constraints of the IoT such as the network traffic, device power, device location etc. The dataset that we use is based on QoS observed over a long time, which represents the average QoS.

For future work, we aim to build a QoS dataset of IoT services, which will include services from WSN, web services and autonomous service providers. This will give a more realistic evaluation compared to our current dataset which is a transformation. We also aim to add time information to this dataset to allow for experimentation with online models. We will conduct more experiments to show how the heterogeneity of services affects the prediction accuracy of these models, which we have replicated for these experiments with a transformation of the dataset. We also aim to investigate a number of alternative techniques for preprocessing the data and conducting similarity comparison between the users (e.g. data smoothing, clustering, PCA, etc.).

ACKNOWLEDGMENT

This work is supported by Science Foundation Ireland (SFI) under the project named SURF: Service-centric networking for urban-scale feedback systems.

REFERENCES


