A Context-Aware Recommendation System with a Crowding Forecaster

Anna Dalla Vecchia[1,†], Sara Migliorini[1,*,†], Elisa Quintarelli[1,†] and Alberto Belussi[1,†]

[1]Dept. of Computer Science, University of Verona, Verona, Italy

Abstract

Recommendation systems (RSs) are increasing their popularity in recent years. Many big IT companies like Google, Amazon and Netflix, have a RS at the core of their business. In this paper, we propose a modular platform for enhancing a RS for the tourism domain with a crowding forecaster, which is able to produce an estimation about the current and future occupation of different Points of Interest (PoIs) by taking into consideration also contextual aspects. The main advantage of the proposed system is its modularity and the ability to be easily tailored to different application domains. Moreover, the use of standard and pluggable components allows the system to be integrated in different application scenarios.

Keywords

Recommendation systems, Crowding forecasting, Deep learning

1. Introduction

Recommendation systems (RSs) are frameworks able to understand and learn users' behaviours by collecting and analysing historical data, with the aim to provide tailored suggestions in a collection of items. These systems are successfully used in many different application domains, from e-commerce and on-demand TV shows, to touristic scenarios. The benefits of producing personalized suggestions are twofold: for the user, choosing an item inside a huge catalogue can become a nightmare, if done without a personalized filtering function; for the service provider, RSs can be used both to build customer loyalty and to guide the user's choice towards a particular subset of items in a given context. With reference to the touristic domain, the ability to guide user choices towards particular attractions in a given context has become urgent in recent years since, due to circumstances such as the COVID-19 pandemic, it has become prominent the need to avoid crowding and restrict the number of people that can access the same Point-of-Interest (PoI) together. A similar requirement can be found also in TV on-demand platforms, where in order to optimize the broadcast service or to promote particular items, there can be the need to suggest new and lesser-known shows with respect to the most popular ones. Therefore, a natural extension of currently available RSs is the ability to predict the level of crowding of a

SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy

*Corresponding author.

†These authors contributed equally.

anna.dallavecchia@studenti.univr.it (A. Dalla Vecchia); sara.migliorini@univr.it (S. Migliorini); elisa.quintarelli@univr.it (E. Quintarelli); alberto.belussi@univr.it (A. Belussi)

0000-0003-3675-7243 (S. Migliorini); 0000-0001-6092-6831 (E. Quintarelli); 0000-0003-3023-8020 (A. Belussi)

given item and to redirect people to other attractions, which are equally appreciated, but are less in demand at the moment [1, 2, 3].

In this paper we present the architecture of a context-aware recommendation system (CARS) [4] with crowd forecasting, which is able to produce tailored suggestions by considering also the expected level of occupation of each item. Relatively to the *crowding forecaster*, we refer to the one presented in [1] and we describe its possible integration inside a CARS. We take as a running example the touristic scenario where we need to suggest the next PoI to visit. However, the system can be easily extended or adapted to other application domains, by properly customizing the notion of crowding and the kind of collected historical data.

The general architecture of the proposed solution is illustrated in Fig. 1. In the first phase represented by box $B1$, the historical user data and the historical contextual data are integrated and processed in order to produce an enhanced dataset of integrated contextual historical data (operation op_1). This new dataset is used as a training input for the deep learning crowding forecaster discussed in [1]. The result of this phase is a model $M1$ which is able to produce an estimation of the level of occupation of a given PoI in a certain context. The trained model $M1$ is then used in a second phase, box $B2$ of Fig. 1, with the aim to produce an estimation of the level of occupation of each PoI in a given context. In detail, at a specific point in time, such level of occupation is computed (or updated) in the background and stored in a database containing also the description of PoIs. All these pieces of information are then made available through a Web Feature Service (WFS) which can feed several different client applications, like a website or a smartphone application (box $B3$).

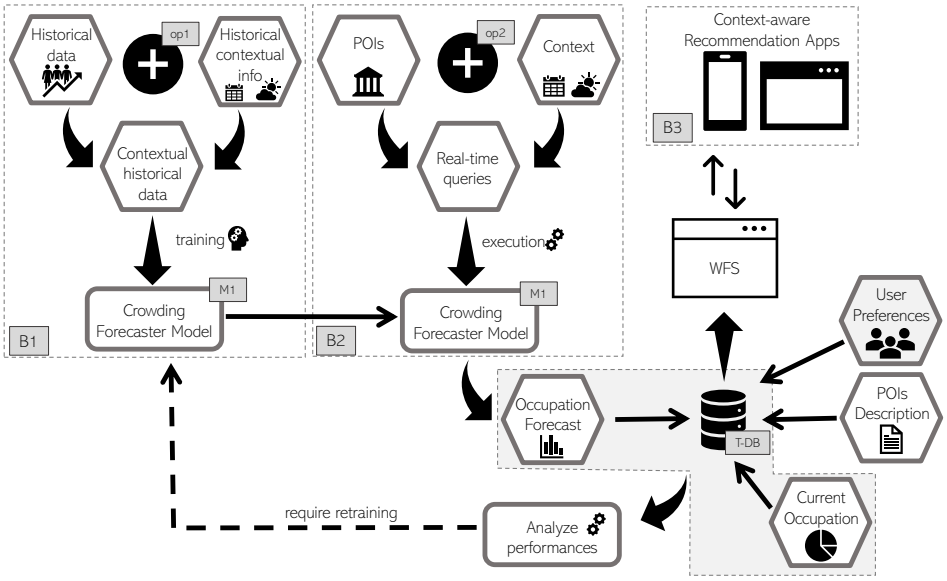

Figure 1: Architecture of the CARS framework.

The remainder of this work is organized as follows: Sect. 2 formalizes the problem, Sect. 3 describes in detail the architecture of the developed system. Sect. 4 illustrates a possible application of the system in the touristic domain and finally Sect. 5 concludes the work.

2. Problem Formulation

This paper considers the touristic domain as the application scenario for our context-aware recommendation system. Therefore, the formulation that follows is tailored for that scenario, but can be easily extended to other domains.

Definition 1 (Touristic visit). *Given a set of POIs $\mathcal{P}$ and a set of users $\mathcal{U}$, a visit performed by a user u is represented as: $v = \langle u, p, t, lat, long \rangle$ where: $u \in \mathcal{U}$ is the user identifier, $p \in \mathcal{P}$ identifies the POI, t is a timestamp representing the date and time of the visit, and lat and $long$ is the spatial position (i.e., latitude and longitude) where the POI is located. The set of all touristic visits performed by users in $\mathcal{U}$ is denoted as $\mathcal{V}$.*

Historical data about past touristic visits can be enriched with some contextual information better characterizing the conditions in which the visit has been performed.

Definition 2 (Contextual information). *Given a visit $v \in \mathcal{V}$, we define its context C as a tuple of values for some relevant dimensions as follows: $C = \langle c_1, \ldots, c_n \rangle$ where each c_i is the value of a contextual dimension cd_i characterizing the problem at hand.*

In our specific scenario regarding the touristic domain, we consider as meaningful contextual dimensions the tuple: $CD = \{ts, doy, dow, hol, pres, wind, rain, temp, hum\}$ where ts is a predefined timeslot inside the day, doy is the day of the year, dow is the day of the week, hol is a boolean value representing the fact that the visit is performed in a public holiday and/or during a weekend, or not, $pres$ is the atmospheric pressure, $wind$ is the wind speed, $rain$ is the amount of precipitation, $temp$ is the temperature, and hum is the percentage of humidity.

Definition 3 (Contextual touristic visit). *Let $v = \langle u, p, t, lat, long \rangle$ a visit performed by a user u in a specific context $C = \langle c_1, \ldots, c_n \rangle$, where $\forall i \in \{1, \ldots, n\}$ c_i is the actual value for the contextual dimension cd_i, a contextual touristic visit is defined as:*

$$cv = \langle u, p, t, lat, long, c_1, \ldots, c_n \rangle \tag{1}$$

where the tuple v representing the touristic visit is enriched with the contextual values in C.

Definition 4 (Crowding forecaster). *A crowding forecaster is a system that once trained with historical data about contextual touristic visits is able to produce an estimate of the level of crowding for a PoI p in a context C.*

The crowding forecaster is an essential ingredient for the development of a context-aware recommendation system.

Definition 5 (Context-aware Recommendation System). *A context-aware recommendation system is a recommendation system which is able to produce useful recommendations by considering not only users' preferences, but also the expected current (or future) level of crowding in the considered set of PoIs.*

3. System Architecture

This section illustrated each component of the proposed framework.

Contextual Data Enrichment

The first operation performed by the CARS framework is the production of the contextual historical data which will be used to train the DL model. This operation is identified as op_1 in Fig. 1. More specifically, besides collecting the past logs about users' choices, it is necessary to identify the sources of information that represent the relevant context for the problem at hand.

In our target scenario, we consider historical data about past touristic visits and we enrich them by deriving some semantic temporal information from the timestamp and by adding information about the weather condition in each specific visit interval. Weather conditions are extracted through the API of OpenWeather [5] at regular and configurable intervals. Clearly, operation op_1 can be customized and enriched with other sources of information based on the problem at hand.

The integration of raw historical data and historical contextual information produces the enriched *contextual historical data* described in Def. 3 and reported in box $B1$ of Fig. 1 as the input for the training of DL model $M1$.

Training of a Crowding Forecasting Model

The crowding forecaster is implemented as a DL model trained with the contextual historical data produced in the previous section. In [1], we tried different machine learning and deep learning models, together with many different configurations. In this paper we assume that the best model identified in [1] is used in box B_1. This is a Deep Neural Network (DNN) implemented and trained in Python by using Tensorflow [6] and Keras [7] libraries. The source code and the datasets used in this paper are available at `https://github.com/smigliorini/crowd-forecaster`.

The trained model is then used to forecast the occupation of each POI in a given context. More specifically, at specific intervals, the future context is retrieved (weather conditions and temporal characterization), and the model is queried in order to produce an estimation of the future level of occupation of each PoI. With reference to box $B2$ in Fig. 1, given the PoIs and the desired context, operation op_2 is responsible to combine them in order to define a query for the forecast model $M1$. The execution of model $M1$ produces a collection of occupation forecasts, one for each PoI in the given context, which is stored in the database T-DB together with other information, like the POI descriptions and the user preferences. As regards to the PoI occupation, the database T-DB is filled not only with the obtained forecast but also with the current (or past) data about occupancy. This information can be made available to users, but it is also useful for evaluating the accuracy of past forecasts and eventually determining the need for a new training of $M1$, if the estimation becomes inaccurate.

Data Publishing via WFS

A Web Feature Service (WFS) is an interface specified by the Open GIS Consortium (OGC) that allows for the exchange of geographic data across the Web [8]. Through a WFS it is possible to share spatial data in standard formats (like GML, GeoJSON, shape files, and so on) and make them available to many different GIS client applications. With reference to Fig. 1, the WFS is used to publish the content of T-DB on the web and make its information usable by a smartphone,

a web or a desktop application. Many different available APIs and desktop applications are compatible with the WFS standard, like Leaftlet [9], QGIS [10], and so on.

Context-aware Recommendation App

Occupation forecasts produced by model $M1$ and stored in T-DB can be used alone or in conjunction with information about the user preferences in order to produce a context-aware recommendation. In Fig. 1, box $B3$ contains different kinds of applications which can use the data stored in T-DB and exposed through a WFS. These applications are called context-aware recommendation apps, since they will produce tailored recommendations by combining user preferences with the expected level of occupation of the PoIs in the given context. In Fig. 1 we abstract from the details of the recommendation system which produces and stores the user's preferences. Clearly, many different tools can be plugged here to obtain this kind of information.

4. Demonstration Scenario

For the demonstration of CARS, we use a real-world touristic dataset regarding the visits performed to a collection of PoIs in Verona, a city in northern Italy. First of all, for the operations in box $B1$ of Fig. 1, we collected about 2,1 million records spanning 6 years (i.e., from 2014 to 2019) regarding the visits performed in 9 different PoIs. Each of these records contains the timestamp and the location of the visited attraction, as well as a category (i.e., Museum, Monument and Church), see Tab. 1 for more details. This historical dataset is enriched with the contextual information regarding the weather conditions and temporal characterization described in Sect. 2. In order to produce a context-aware recommendation, CARS needs also to know the preferences of each user with respect to the various POI categories in the dataset. At this regard, we collect the preferences of a set of users and store them in the database.

Table 1
Historical dataset for operation op_1.

Name	Category	Num. of visits
Arena Amphitheatre	Monument	421,490
Juliet's House	Museum	375,305
Lamberti's Tower	Monument	290,243
Castelvecchio Museum	Museum	271,552
Church of St. Anastasia	Church	230,352
The Cathedral	Church	205,293
Roman Theatre	Museum	145,854
Palazzo della Ragione	Monument	111,440
Juliet's tomb	Museum	100,701

Given the trained model $M1$, we use a Python script to periodically query the model for each PoI and in a desired temporal and weather context. The obtained results are stored in a PostgreSQL database with PostGIS extension. Fig. 2 illustrates the interface of the GeoServer (https://geoserver.org/) tool used to configure the WFS and expose the database content.

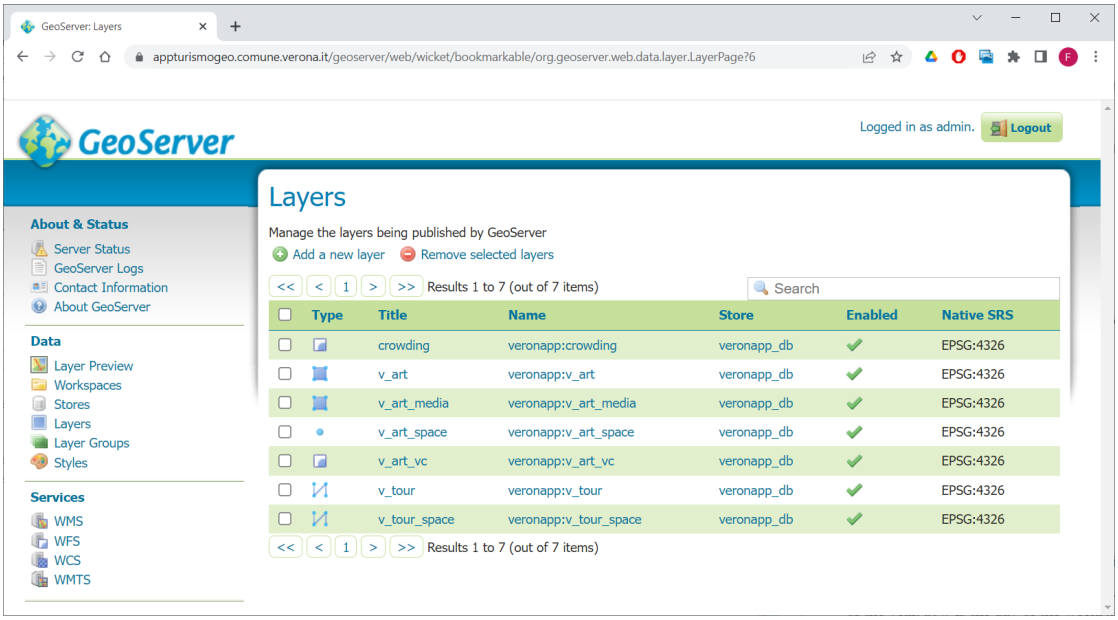

Figure 2: Set of layers provided by a GeoServer WFS.

In order to illustrate the system functioning, we implement the recommendation application illustrated in Fig. 3. Each authenticated user can see his/her position on the map and can obtain a suggestion for the next PoI to visit based on his/her preference for the various attraction categories and his/her proximity of the various PoI. More specifically, given a user $u \in \mathcal{U}$, the preference of u for a PoI $p \in \mathcal{P}$ is computed as

$$P(u,p) = P(u,c) \cdot \frac{1}{\delta(p,u)} \cdot \frac{1}{(1+\#p)} \tag{2}$$

where $\delta(p,u)$ measures the spatial distance between the POI p and the user u, while $P(u,c)$ is the preference of the user u for the category c of p, and $\#p$ is the number of times the user u has already visited p in the past. The spatial distance is computed by considering the road network and using the libraries OSMnx [11] and NetworkX [12]. From the equation we can observe that the preference for p increases as the preference for the category c of p increases, and it decreases as the distance between the current position of the user and the PoI location increases, or with the number of times user u already visited p in the past.

Given such definition of preference of a user u for a PoI p, we can also introduce the expected level of crowding produced by the model $M1$ in the final computation. Therefore, the contextual preference of the user u for the PoI p in the context C becomes:

$$P^*(u,p,C) = P(u,p) \cdot \left(1 - \frac{ev(p,C)}{cap(p)}\right) \tag{3}$$

where $P(u,p)$ is the static preference defined in Eq. 2, while $cap(p)$ is the maximum capacity of the PoI p, namely the maximum number of visitors it can host and $ev(p,C)$ is the estimated number of visitors determined by the crowding forecaster $M1$ for PoI p in the context C.

The web application illustrated in Fig. 3 allows us to demonstrate the potentiality of the framework and in particular to compare the different suggestions produced by using either Eq. 2 or Eq. 3, namely by considering only the users' preferences or also the level of crowding. The general interface of the application consists on a web map where the user's position is

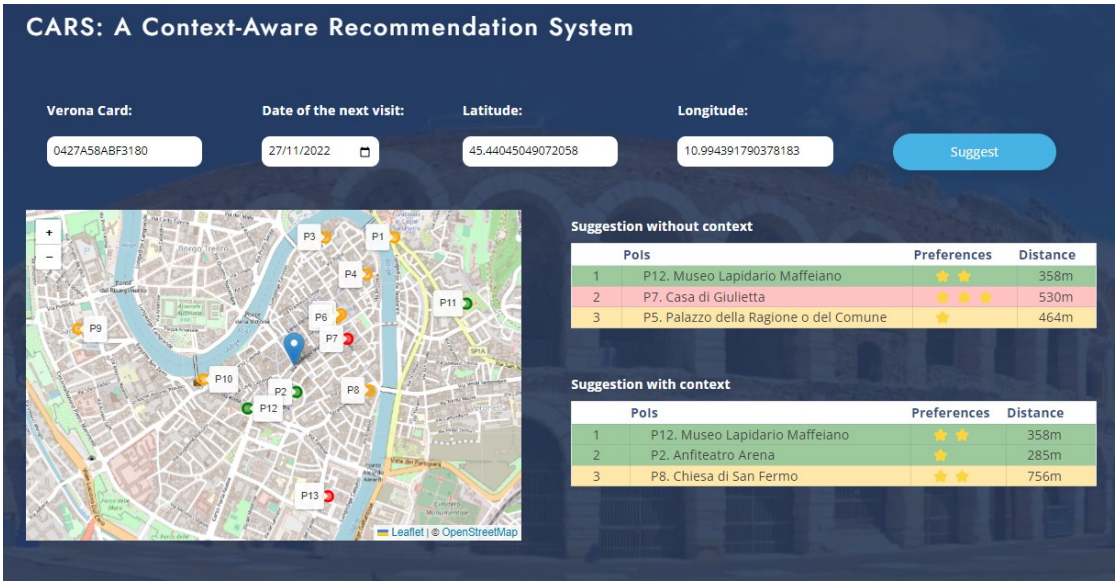

Figure 3: CARS web application

highlighted with a blue placeholder, while the location of each PoI is denoted with a placeholder whose color represents the different levels of occupation: green means quite empty, yellow means normally occupied, while red denotes an overcrowded situation. At the top of the map, there are some input fields that allow the user to specify the spatial and temporal context of the visit: the user identifier and position, and the desired date for the visit. The user position can be specified both textually or by performing a selection on the map. Finally, the button *Suggest* applies Eq. 2 and Eq. 3 for producing the static and contextual recommendation, respectively. The system returns two lists with the first three best PoIs for the user: each row has a color depending on the level of occupation of the PoI in the given date, and it reports the preference of the user for the PoI category as a set of stars, from 1 to 3, as well as the distance of the PoI from the current user position.

As an example we report in Tab. 3 the different suggestions produced in two different contexts $C1$ and $C2$. More specifically, for $C1$ we choose a sunny day during the weekend, while for $C2$ we consider a sunny Thursday, which is one of the quieter days in terms of tourist visits. The level of occupation of each PoI in the two contexts is described in Tab. 2. As you can notice in $C1$ there are some PoIs, like P7 and P13, that are overcrowded, while others like P2 and P11 that are average occupied. Conversely, in C2 all occupation rates are below 50%.

Tab. 3 shows different static and contextual recommendations suggested to two users $U1$ and $U2$ in these two contexts. Besides to the three suggested PoIs, it reports the distance of each PoI from the user position and the user preference for the PoI category (3 is the maximum

Table 2
PoI occupancy rates for two contexts

Context	P1	P2	P3	P4	P5	P6	P7
$C1$	75%	42%	66%	66%	75%	75%	83%
$C2$	50%	42%	33%	42%	50%	42%	42%
	P8	**P9**	**P10**	**P11**	**P12**	**P13**	
$C1$	58%	75%	66%	50%	50%	83%	
$C2$	50%	42%	33%	50%	42%	25%	

value and 1 is the minimum value). We can notice that in context $C1$, PoI $P7$ has a level of occupation close to saturation; therefore, even if $U1$ has a greater preference for it, the system suggests $P2$ in place of $P7$. Similarly, for $U2$ even if $P7$ is spatially closer to user's position, it does not appear in the contextual suggestions. Conversely, in context $C2$, PoIs are typically unloaded and the set of suggestions is the same for both static and dynamic approaches.

Table 3
Result of recommendations

Cont.	User	Static Rec.			Contextual Rec.		
		PoI	Distance	Pref.	PoI	Distance	Pref.
$C1$	U1	P12	358m	2	P12	358m	2
		P7	530m	3	P2	285m	1
		P5	464m	1	P8	756m	2
	U2	P12	358m	3	P12	358m	3
		P2	285m	2	P2	285m	2
		P7	530m	3	P10	619m	3
$C2$	U1	P12	358m	2	P12	358m	2
		P7	530m	3	P7	530m	3
		P5	464m	1	P5	464m	1
	U2	P12	358m	3	P12	358m	3
		P2	285m	2	P2	285m	2
		P7	530m	3	P7	530m	3

5. Conclusion

In this paper we have presented the possible architecture of a context-aware recommendation system enriched with a crowding forecaster. This system is able to produce a set of dynamic suggestions to users where the preference associated with each item depends also on its level of occupation in different contexts. The use of a contextual crowding forecaster allows producing more precise estimations and more personalised and useful suggestions. The system is currently under experimentation as a prototypical touristic application developed in conjunction with the Touristic Office of Verona, in Italy.

References

[1] A. Belussi, A. Cinelli, A. Dalla Vecchia, S. Migliorini, M. Quaresmini, E. Quintarelli, Forecasting POI occupation with contextual machine learning, in: Proc. of the 26th European Conference on Advances in Databases and Information Systems, ADBIS, 2022, pp. 361–376.

[2] S. Migliorini, D. Carra, A. Belussi, Distributing tourists among POIs with an adaptive trip recommendation system, IEEE Transactions on Emerging Topics in Computing 9 (2021) 1765–1779.

[3] S. Migliorini, D. Carra, A. Belussi, Adaptive trip recommendation system: Balancing travelers among POIs with MapReduce, in: 2018 IEEE International Congress on Big Data (BigData Congress), 2018, pp. 255–259.

[4] N. M. Villegas, C. Sánchez, J. Díaz-Cely, G. Tamura, Characterizing context-aware recommender systems: A systematic literature review, Knowl. Based Syst. 140 (2018) 173–200. doi:10.1016/j.knosys.2017.11.003.

[5] OpenWeather, 2023. URL: https://openweathermap.org/api.

[6] M. Abadi, et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL: https://www.tensorflow.org/.

[7] F. Chollet, et al., Keras, 2015. URL: https://github.com/fchollet/keras.

[8] A. Sinha, Web Feature Service (WFS), in: S. Shekhar, H. Xiong (Eds.), Encyclopedia of GIS, Springer US, 2008, pp. 1256–1259. doi:10.1007/978-0-387-35973-1_1479.

[9] Leaflet - an open-source JavaScript library for mobile-friendly interactive maps, 2023. URL: https://leafletjs.com.

[10] QGIS A Free and Open Source Geographic Information System, 2023. URL: https://www.qgis.org.

[11] G. Boeing, Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks, Computers, Environment and Urban Systems 65 (2017) 126–139.

[12] Networkx, 2014. URL: https://networkx.org.