Water-COLOR: Water-COnservation using a Learning-based Optimized Recommender

GuangXue Zhang*, David L. Feldman*, Yiming Lin§, Sharad Mehrotra*,
Nalini Venkatasubramanian*, Thayer Drew†, Kim Sentovich†, Owen Veranth‡
*University of California, Irvine.
{gxzhang7, feldmand}@uci.edu {sharad, nalini}@ics.uci.edu
§University of California, Berkeley.
yiminglin@berkeley.edu
†Rachio Inc.
{Drew.thayer, kim.sentovich}@rachio.com
‡Analytiks Inc.
owen@analytiks.co

2024 IEEE International Conference on Smart Computing (SMARTCOMP) | 979-8-3503-4994-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/SMARTCOMP61445.2024.00034

***Abstract*—Efficient water use, particularly in the realm of irrigation, has emerged as a critical concern in regions suffering from persistent drought, such as California and Florida. With the advent of smart irrigation controllers encouraged by environmental policies, a new paradigm of water management is gaining traction. Among these, the Rachio smart controller has garnered significant attention. However, without direct feedback or actual water usage data, optimizing these irrigation systems for enhanced efficiency remains challenging. This paper introduces Water-COLOR, a novel recommendation system integrated within the Rachio smart controller's framework to address this challenge. The system leverages similar landscape profiles to suggest irrigation schedules that are both water-efficient and user-preferable. By analyzing manual user interactions with the controller, Water-COLOR infers user satisfaction, which, along with estimated water usage, informs the adaptation of irrigation plans. The system eschews the need for additional sensors, thereby reducing infrastructure requirements. Our evaluation demonstrates consistent performance across diverse climatic regions and indicates that the system's recommendations could significantly contribute to water conservation efforts. The results not only showcase the potential of Water-COLOR to enhance the efficiency of existing smart irrigation systems but also open avenues for deploying real-time, data-driven environmental solutions.**

I. Introduction

Over the past two decades, as urban areas continue to expand, irrigation water systems serving housing with lawns and yards have rapidly been developed and deployed [6]. In the United States, this growth is particularly notable in states like California and Florida, where prolonged drought conditions have persisted for years [8]. In these regions, reducing irrigation demand is an effective strategy for conserving potable water [7]. Recognizing the importance of efficient outdoor water use, the state of California has actively encouraged the replacement and installation of smart irrigation controllers in landscaping through rebate programs [9]. Furthermore, an investigation [10] involving 838 single-family homes across the U.S. revealed that irrigation accounts for half of their total water consumption.

In 2011, there were a few choices of commercial controllers available in the market, whereas in 2020, there are more than 700 controller models with EPA WATER-SENSE Label [11]. Irrigation controller first improved from manual operation to an automatic controller, named "Timer", that starts and stops irrigation based on preset fixed schedule, e.g fixed days in a week, starting time and duration time. Timer relieves humans from having to interact with the controller, but according to [12] homes with a traditional timer used 47% more water than homes without. Essentially, timers just brought convenience to people, but lacked any consideration of irrigation efficiency.

Therefore, homeowners looking for potential water bill savings need smart controllers to conventional systems with a traditional timer. Sensor-based controllers are coming to market, with Soil Moisture Sensors (SMS) being a representative example [13]. Soil moisture sensors are installed underground in the root zones and transmit moisture data to the controller. This allows for irrigation schedules to be triggered only when moisture levels fall below a threshold. Based on this idea, scholarly research [17], [18] has extended to other sensor types, such as those measuring temperature, humidity, and using webcams to monitor plants/vegetation. Such systems often leverage PID (Proportional, Integral, Derivative) in feedback control systems or ANN (Artificial Neural Network) models to devise an optimal irrigation schedule, aiming for efficient water use while maintaining plant health [16].

Another type of controller called weather-based or ET-based has become popular in the market due to the recognition of evapotranspiration (ET) as a critical factor for determining when to irrigate [13]. ET is the combination of evaporation from the soil plus transpiration from plants and depends on weather factors such as sunlight, temperature, wind, and humidity [15]. ET-based irrigation controllers adjust irrigation events based on historical or on-site weather data.

Although SMS based irrigation controllers can be more effective according to [10], it requires additional infrastructure and investment, making it impractical for installation in every household. ET-based controllers, in contrast, get rid of sensors

2693-8340/24/$31.00 ©2024 IEEE
DOI 10.1109/SMARTCOMP61445.2024.00034

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 25,2024 at 07:37:24 UTC from IEEE Xplore. Restrictions apply.

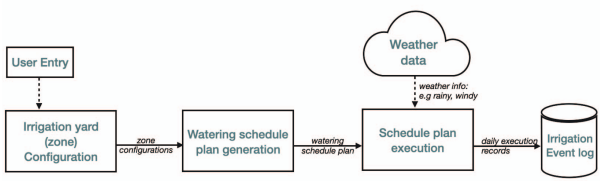

Fig. 1. Rachio irrigation smart controller workflow

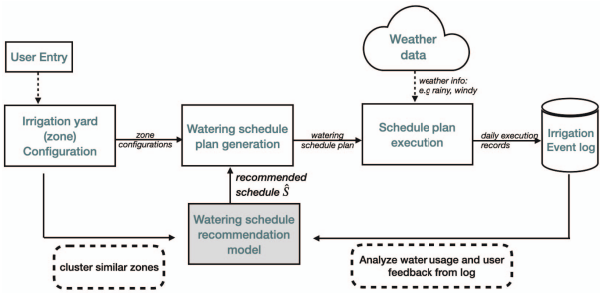

Fig. 2. Rachio irrigation smart controller integrated with recommendation model workflow

making them a more viable option. However, ET-based controllers often rely on user input for landscape configuration for correct crop type, soil type etc. If the input information is not accurate, the ET coefficient calculation is not accurate, making the controller not optimal for water efficiency. Using ET controllers requires users to have some degree of agricultural knowledge or experience, or to be educated by a smart controller provider or local district water agencies. Meanwhile, if the homeowners experience challenges while using the smart controllers, their level of satisfaction with the technology degrades [6]. Consumer feedback-based designs haven't been much considered in the current smart controller market, but understanding their preference and feedback helps to create more effective water conservation programs [14].

To address the above challenges and issues, we have developed a recommendation system for an ET-based smart irrigation controller aimed at enhancing irrigation efficiency. Without direct feedback data such as water usage data and soil moisture level data, we bring the idea from recommendation models to match similar landscape houses and recommend preferable and water effective schedules. Thus, our system does not require additional sensors to be installed, allowing for sharing effective irrigation schedules from both experienced and agriculturally educated users. Moreover, we consider user satisfaction by analyzing the frequency of manual interactions with the controller. This data is used to refine and adjust the irrigation schedules, ensuring they not only conserve water but also meet the user's needs and expectations more accurately.

The rest of the paper is organized as follows: in section II, we list related works on smart irrigation controllers designs, and basic recommendation models as well as their applications. In section III, we present the problem setting with data schema descriptions, assumptions, and problem formulation. Section IV presents the system architecture and details about data flow and algorithms we choose for each module. Section V contains the evaluation metrics and experiment results. Finally, the conclusions and potential future work are discussed in Section VI.

II. Workflow

A. Rachio's Controller Workflow

We implement our recommendation with the Rachio smart irrigation controller. Rachio makes the top-rated smart sprinkler controller in the United States nation [41]. It has a customer base of over one million users and has won many prestigious awards. In this section, we will first describe the original workflow of Rachio's smart controller to understand its functionality. Following that, we explain our architecture to integrate the recommendation model into this workflow.

The original workflow is shown in Figure 1. After installing the irrigation controller and wiring the sprinklers in the yard, users can segment their yard into limited numbers of zones (general max number: 8 or 16, according to device model) based on distinct characteristics like plant and soil types. Each zone can then be configured within the Rachio app. Following the zone configuration, the app will generate a watering schedule plan accordingly, which contains the watering dates and starting times, and durations, along with other tunable parameters. If the user select a FLEX type of schedule, the app accesses weather data from either cloud services or local weather stations to smartly calculate the plant need according to ET demand and build an irrigation schedule to most efficiently water the plants. All Rachio schedules can also skip irrigation during rainy, windy, or snowy days. Additionally, users can always manually start or stop the controller for immediate irrigation or stop an ongoing watering event. All related data, including records of each operation, is uploaded to Rachio's cloud platform.

Our goal is to recommend for users irrigation schedules derived not just from zone configurations but also from an intelligent recommendation model, to enhance water efficiency. Our solution is to integrate a recommendation model into the Rachio workflow as shown in Figure 2. Using the zone configuration and irrigation log as input, our recommendation model can benefit other similar user's schedules and recommend less-experienced users schedules with better water efficiency.

B. Data Description

The user begins with defining and configuring each zone by specifying basic yard features, which are saved in the *zone detail* table. Those features include zone size, crop type, soil type, crop root depth, nozzle type etc. Those features will determine how much water the yard needs, and the system will provide a watering schedule based on the features. Customers lacking knowledge of their zone configuration can choose to leave every setting as a default value, which leads the system to provide a basic but potentially non-optimal watering plan. Irrigation watering plan is called schedule and its attributes are

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 25,2024 at 07:37:24 UTC from IEEE Xplore. Restrictions apply.

stored in the *schedule detail* table, which has the schema as {schedule type, the days of watering, start time, duration}. The *schedule type* attribute includes FIXED and FLEX categories. The FIXED schedule operates independently of daily weather conditions, executing irrigation on predetermined dates, such as every two days and disabling the skip operation. In contrast, the FLEX schedule takes weather data into account, creating a schedule to optimally water according to ET demand. Users with FIXED or FLEX schedules can opt to skip watering in response to weather conditions, such as rainy days. Users can also adjust the schedule by manipulating those parameters. Event datasets record customer's daily irrigation log for each enabled zone. The log includes start time and end time of each scheduled irrigation, as well as all user's manual operations. The user operations involve manually triggering or stopping irrigation, affording users the ability to instantly water their garden. In the later case, the irrigation will be labeled as *manual*, otherwise *schedule*.

III. System Architecture

A. Assumptions

Our goal is to enhance water efficiency for irrigation by recommending to users one or more schedules suitable for the zone, which are optimized for water conservation. Two questions arise: First, for a given user, which watering schedules should be considered as candidates for recommendation? We note that similar zone configurations often share similar irrigation schedules, as the needs of plants in these zones are likely to match closely. We can also augment the zone configuration to include features such as location and ET (evapotranspiration) value. Thus, the candidate set of recommendations correspond to those that are being used by other users with similar zones/landscapes.

Note that candidate schedules identified based on zone similarity might not be suitable for a zone since it may result in under-watering or over-watering of plants. This leads to the second question - of all the candidate schedules, how to determine which schedules are suitable for the zone/landscape of a given zone? To address the question, we make an assumption that most users equipped with a smart sprinkler controller are not willing to have their plants over or under watered. Therefore, they will interact with the controller if they are not satisfied with the current scheduled irrigation plan. That being said, when plants are under-watered, users will interact with the schedule by either updating it or performing manual watering operations as needed. Conversely, if plants are receiving excessive water, users are expected to manually halt the scheduled irrigation for that day. Either way they will leave a record in the event data for manual operations. By counting the number of user interactions and/or duration of total interactions for each month, we can interpret that, the more interactions with the controller, the less the user is satisfied with the current irrigation plan. While this assumption might not be true for all users, it does reflect a reasonable expectation of why users would manually control their system. Customers who have yards or gardens and are willing to invest in a smart irrigation controller typically have interest in ensuring the health and vitality of their plants. The ideal case is to have an optimal irrigation plan so that users are hand-off for the irrigation task.

Finally, for all the suitable schedules which we can recommend, we need to address which one we should recommend. Since our goal is water conservation it is natural to choose a schedule based on estimating the water usage for the given schedule.

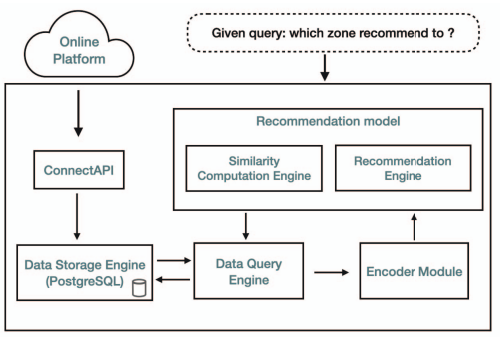

Fig. 3. System Architecture

B. System Architecture Overview

In Figure 3, we provide an overview of the system architecture, with a more detailed examination of each module and our methodologies to follow in subsequent sections. The system starts with retrieving recent data from the cloud platform via a ConnectAPI, which outlines the schema of the data to be fetched and the event data over a specified time interval. This data is then stored in a local PostgreSQL database [2], utilizing the same table schema as that used on the cloud platform. Given a target zone specified by users, the system aims to recommend a water-efficient schedule plan. To this end, the system first identifies the top-k zones most similar to the target zone. However, since zones in raw data may include non-numerical attributes, an encoder module is required to convert the non-numerical data into numerical vectors to facilitate later similarity computation. The similarity computation engine identifies the top-k most similar zones as candidates. Recommendation engine is responsible for recommending the best schedule from the candidates based on the execution statistics in current watering plans and the estimation of water usage per zone area (square feet) from the event data in the last six months.

C. Architecture Component Detail

a) ConnectAPI to cloud platform: IoT devices generate data that is saved to cloud platform. In our implementation, the system is built locally and thus a data sourcing module is required to allow downloading and updating data from the cloud platform. There are two reasons, first, network latency makes it impractical to query and process complex models in real-time online. Secondly, it allows us to selectively store

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 25,2024 at 07:37:24 UTC from IEEE Xplore. Restrictions apply.

essential datasets (such as the most recent six months of event data from years of accumulated data, and zone data specific to users in a targeted area like California) rather than the entire United States, which further reduces the processing latency. The *ConnectAPI* module allows access to the cloud platform under authorization to fetch data using SQL query. The data is saved and stored in a local PostgreSQL database.

b) Encoding Module: As the raw data of zone configuration contains non-numerical values (e.g., the category value $sand$ under attribute zone soil type), we implement a encoding module to transform non-numerical values into numerical values in the zone configuration vector. The transformation relies on a set of rules specified by agriculture experts. In particular, each rule defines a one-to-one mapping from a category-type attribute to a numerical-type attribute. For example, the attribute zone soil type is mapped to a numerical attribute called water retention capacity, and the $sand$ value is mapped to 0.05 denoting the water retention capacity for different types of soil. We further normalize the transformed numerical values such that they are in the same range to facilitate later similarity computation. For instance, one feature in zone configuration called *zone_root_zone_depth* spans the range [101.6, 381.0], whereas another feature called *infiltrationRateValues* spans the range [0.0007, 0.004]. We take the normalization formula as below to avoid this problem:

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)}, \tag{1}$$

where a and b are the desired lower bound and upper bound after normalization, in our case, we assign $a = 0.1, b = 1$.

c) Similarity Computation: The similarity computation module takes normalized zone configuration vectors as input, for a given target zone vector, generates top-k most similar zone vectors.

Given two encoded zone vectors z_1 and z_2, we compute the cosine function value as their similarity score, denoted by

$$Sim(z_1, z_2) = \frac{z_1 \cdot z_2}{\|z_1\|\|z_2\|}. \tag{2}$$

The cosine value, which falls within the range of $[0, 1]$, can be interpreted as a similarity score. A similarity score closing to 1 signifies that two vectors are very similar or closely aligned.

A standard method for computing similarity using the cosine function involves creating a cosine similarity matrix, as mentioned in [35]. By retrieving the i_{th} row (or the i_{th} column, due to the matrix's symmetry), one can obtain the cosine similarity values between the i_{th} element and all other elements. However, this straightforward approach to computing cosine similarity can incur significant computational and storage cost when dealing with large input sizes. (e.g., in our case, the number of zone vectors) For instance, computing a cosine similarity matrix for 40,000 vectors (where each vector has 20 dimensions) on a personal laptop (e.g., Apple M2 chip, RAM 16GB) takes about 10 seconds. The computational cost jumps to around 40 seconds when the number of vectors with the same dimension increases to 50,000, observing a significant quadratic increasing of computational cost with vector size. Additionally, large vector size also incurs significant resource consumption due to a potential large similarity matrix. Moreover, as the need arises to include more zone tuples in the future, the matrix size will expand quadratically, which makes the similarity matrix impractical to be stored in memory. To address this challenge, we store the vector of zones in vector databases [36] to speed up similarity computation and save storage cost. In particular, vector databases directly store a set of vectors instead of the large similarity matrix, and they support index-based fast similarity computation [37] under various similarity metrics, such as cosine similarity, Euclidean distance, or L2 norm. In our implementation, we use the PostgreSQL extension *pgvector* [39] to store vectors and compute similarities. We employ the HNSW [38] graph-based indexing strategy, currently recognized as the state-of-the-art method.

d) Schedule Metric: In a general recommendation system such as online shopping or movie recommendation system [40], user feedback is often captured by rating scores of products or movies, reflecting their satisfaction with the item. Unlike these systems, our watering schedule recommendation system doesn't have a direct rating mechanism provided by users. Therefore, we use a satisfaction score over usage per unit ratio to qualify a schedule, which serves as rating score in our recommendation system, allowing us to assess the quality of each watering schedule.

Satisfaction Score (sc): We define the monthly user satisfaction score on a schedule S by counting the numbers of scheduled operations divided by the numbers of total irrigation operations, including scheduled and manual operations for each month:

$$sc(S) = \frac{\#scheduled}{\#scheduled + \#manual}. \tag{3}$$

If the schedule is applied to more than one zone, we average the satisfaction scores among all applied zones. The definition of satisfaction score (sc) in Equation 3 is consistent with the assumption we made in Section III.A, indicating that more user interactions and manual operations suggest a lower user satisfaction with the existing irrigation schedule plan.

Schedule Quality: We use the ratio between user satisfaction score (sc) and the estimated water usage per unit to define the schedule metric for a given schedule S:

$$Q(S) = \frac{sc(S)}{\sum_{days \in month} duration(day)/z_i.area}, \tag{4}$$

where Q denotes the quality, and *duration* is each irrigation duration time, regardless of manual or scheduled operation, that is saved in minutes, and *area* is one given zone z_i's area in square feet. The denominator aggregates the total irrigation time over a month, to overcome the bias of different month, we takes into account for up to last six month's records and average it to get the quality of a given schedule S. If one schedule has been created recently, e.g has been used for one or two months, we can eliminate the case and search for other longer candidates.

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 25,2024 at 07:37:24 UTC from IEEE Xplore. Restrictions apply.

e) Recommendation Model: The recommendation module takes the input of the top-k similar candidate zones, fetches out those zones' current schedules, computes each schedule's quality using Equation 4, and outputs a recommended schedule, noted by $\hat{S}$. The process for recommendation is shown in Algorithm 1.

Taking the input with candidate schedule lists and quality lists of those schedules, we introduce a Correlated Weighted Majority Vote(CWMV) algorithm as the recommendation strategy. For a tuple of schedule S, it consists of a list of attributes $\{s_i, \cdots, s_m\}$, where $m = 16$ but can be varied according to different feature selection methods. The schedule attributes are categorical data type or boolean type, in which the domain values for each attribute are finite. Therefore, generating a recommended schedule, denoted by $\hat{S}$, is equivalent to assigning a specific value (from possible domain values) for each attribute. To determine the optimal value for each attribute, we use weighted majority vote on each attribute among the schedules in the candidate zones. The weight coefficient of each schedule S is defined by $Q(S)$ in Equation 4. However, the majority voting mechanism needs to consider the correlated relationship between attributes. If there exist correlated attributes, the majority vote mechanism should take them as a group and count on the group values. Therefore, a correlation analysis for schedule attributes needs to be done before-hand.

To capture the correlation, we use phi-coefficient between two attributes [1]:

$$\phi_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{5}$$

where ϕ is the correlated coefficient in the range [-1,1]. If $\phi = -1$ or $\phi = 1$, it indicates a strong negative or positive correlation relationship between those two attributes, respectively.

Based on the correlation analysis, attributes that are correlated can be grouped together, treating their possible domain values as a collective group value. The Correlated Weighted Majority Vote (CWMV) approach is outlined in Algorithm 2. For each schedule attribute s_i (or a group of correlated attributes), a Python dictionary is constructed. This dictionary is then used to iterate over the list of schedules from candidate zones, logging each encountered domain value and accumulating a sum by the quality of the schedule $Q(S_j)$, which serves as the weight for that domain value. Upon completing each iteration, the domain value with the highest total weight is selected. By determining the value for each attribute, we generate the recommended schedule $\hat{S}$ as output of CWMV.

Algorithm 1: Recommendation Algorithm

Input: z_i
candidate K zones: $\{z_1, \cdots z_k\} \leftarrow$ similarity(z_i, k)
$candidateSchedulesList := []$
$candidateSchedulesQualityList := []$
for $z_j \in \{z_1, \cdots z_k\}$ **do**
  Fetch z_j's schedule S_j
  $candidateSchedulesList$.append(S_j)
  Compute $Q(S_j)$
  $candidateSchedulesQualityList$.append$(Q(S_j))$
$\hat{S} \leftarrow CWMV(candidateSchedulesList,$
$candidateSchedulesQualityList)$
Return $\hat{S}$

Algorithm 2: CWMV: Correlated Weighted Majority Vote

Input: $SchedulesList, SchedulesQualityList$
for *attribute* $s_i \in \{S.s_1, \cdots, S.s_m\}$ **do**
  $AttributeValue := \{value : weight\}$
  for *index* j *in range* $SchedulesList$ **do**
    $S_j.s_i \leftarrow SchedulesList[j].s_i$
    if $S_j.s_i$ *not in* $AttributeValue$ **then**
      $AttributeValue[S_j.s_i] = 0$
    $Q_j \leftarrow SchedulesQualityList[j]$
    $AttributeValue[S_j.s_i] \mathrel{+}= Q_j$
  $\hat{S}.s_i \leftarrow max(AttributeValue[S_j.s_i])$
Return $\hat{S}$

IV. Evaluation

To assess a recommendation system effectively, it is essential to have accurate ground truth data, indicating if the recommendations align with expectations. However, in this application setting, we lack precise ground truth data, including actual irrigation usage and user feedback on the smart irrigation controller. Obtaining household level data and feedback at the community scale is infeasible; smart controllers (e.g. Rachio device) utilize evapotranspiration (ET) estimate to assess household water usage at desired levels of granularity. Detailed information would require homeowners to install additional hardware/equipment to measure volumetric flow in the outdoor irrigation system. Our methodology is to choose a subset of data (zones) that can serve as representative ground truth data with the designed constraints. In the following sections, we will first outline our evaluation metrics under the assumption that ground truth data is available. Then, we explain our method for selecting a subset of data to serve as a stand-in for ground truth. Lastly, we will present our experiment results.

Designing Evaluation Metrics for Household Water Conservation: For a given zone z_i, an ideal schedule or ground truth schedule is denoted by S^* - this ideal schedule is expected to achieve highly positive user feedback and a reasonably low irrigation usage. Generally speaking, S^* is the ideal schedule that users can rely on; it allows users to operate the daily irrigation task in a hands-off manner. We denote this ideal schedule set by D_{ideal}

a) Precision: We next define the precision of a recommendation schedule $\hat{S}$ wrt the ideal schedule S^* as follows:

$$\text{Precision}(\hat{S}) = \frac{|S^* \cap \hat{S}|}{|S^*|}. \tag{6}$$

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 25,2024 at 07:37:24 UTC from IEEE Xplore. Restrictions apply.

This precision represents the proportion of schedules where $\hat{S}$ has an intersection with S^*. More precisely, each attribute of the schedule takes on a discrete value - the type of this value may be float, string or boolean. The intersection in the above equation indicates the number of attributes where the recommended schedule and ideal schedule have the same value. The overall precision is thus calculated as :

$$\text{Precision} = \frac{1}{|D_{ideal}|} \sum_{S_i \in D_{ideal}} \frac{|S_i^* \cap \hat{S}_i|}{|S_i^*|}. \quad (7)$$

b) Accuracy: We next define the accuracy by calculating the mean square error between quality of a recommended schedule $\hat{S}$ and an ideal schedule S^*. The overall accuracy for is as follows:

$$\text{RMSE} = \frac{1}{|D_{ideal}|} \sqrt{\sum_{S_i \in D_{ideal}} (Q(\hat{S}) - Q(S^*))^2}. \quad (8)$$

A. Prepare Ground truth data

We select schedules as members of D_{ideal} according to following constraints:

- The schedule has to belong to an active user's zones. Active user means having event records in the last 180 days.
- The schedule associated zone configuration containing more non-default values are preferable, since users are willing to customize their zone configuration are assumed to be more active to the irrigation controller, therefore their satisfaction scores sc are closer to the real situation.
- Our goal is to recommend an ideal schedule S^* that should has equal or better quality than the current one. Thus, we should select ground truth data only with ideal schedules, i.e high quality schedules with high satisfaction score and reasonably low irrigation usage. Therefore, experiment result should show that the recommended schedule $\hat{S}$ is close to S^* in terms of evaluation metric.

B. Experimental Results

*a) **Precision and MSE vs. Similarity methods**:* In total, we chose 96398 numbers of zones around the Orange County, California area, and we selected 9221 zones as a testing dataset. Assume that their schedule is ground truth schedule S^*, and we recommend a schedule $\hat{S}$ for each of the zone in the testing dataset. For comparison, we also test the zones in other areas that have different weather conditions and precipitation patterns than the state of California. We selected Seattle, Washington, which is one of the most rainy cities in the US. We also select cities near Orlando, Florida, which conversely to California, is dry in winter and humid in summer season. The precision and MSE result is shown in Table I.

We also test with different similarity methods other than the default cosine similarity. We choose the L2 norm, also known as Euclidean distance, and inner product to capture the similarity between zone vectors. The results of L2 norm is shown in Table II, and the results of inner product is shown in Table III.

In interpreting the experiment's results, particularly the precision metric hovering around 0.53, it's crucial to consider the nature of the output and ground truth within the context of the system. Precision in our experiment is defined by the ratio of intersection between the model's output and a human-selected ground truth. Given that both consist of tuples with 16 attributes characterized by boolean or categorical data, the likelihood of an exact match decreases due to the combinatorial complexity of the attribute space. The first point to consider is the presence of attributes that may be "None", which does not necessarily reflect a predictive error but rather a gap in the available data. Secondly, the high number of possible values for certain attributes, such as *Schedule Cycle Time*, naturally makes exact matches less likely. Furthermore, the ground truth, being a product of human selection, may not necessarily represent the actual reality, as the real ground truth is inaccessible.

In terms of different dataset, from the three tables result we can observe that, our recommendation system performs equally well in different dataset, and being consistent for different location and weather users.

*b) **Precision and MSE vs. Top-k coefficient**:* The numbers of nearest neighbors of given zone is denoted by k, and it determines the number of candidate zones we select. Choosing an optimized k can have impact on the recommendation system performance. If k is too small, e.g $k < 10$, the Correlated Weighted Majority Vote can not provide a high confidence result due to lack of enough voting members. On the other hand, if k is too large, e.g $k > 200$, one may include too many candidate zones, and the fact of majority default settings may impact on the Correlated Weighted Majority Vote algorithm and thus the default setting may poison the vote result. In addition, large k can also lead to longer latency, due to the fact that the system has to analyse more event data.

We use the cosine similarity on three cities testing dataset, and vary the top-k coefficient in set $\{10, 30, 50, 80, 100, 150, 200\}$. From the precision experiment varying with k coefficient, the result in Figure 4(a) has shown that, precision initially rises with an increase in k, but starts to decline if k continues to increase. This observation aligns with the analysis that was described previously.

The result in Figure 4(b) shows a clear decrease trend of Mean Squared Error (MSE) as k increases. MSE is the metric that evaluate how the recommended schedule close to the selected ground truth schedule through a regression model. As k increases, more candidate schedules can be included for training to refine the regression model, and thus the error decreases. However, when k exceeds 50, the result implies that adding more schedules does not significantly enhance the model's performance, as the error rate approaches a stable minimum.

*c) **Baseline: Precision and MSE vs. Top-k coefficient**:* To demonstrate the performance of our recommendation system, we conducted an experiment comparing it to a baseline method. Since there are no existing studies using recommendation models or learning-based algorithms for irrigation

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 25,2024 at 07:37:24 UTC from IEEE Xplore. Restrictions apply.

Cities	testing zones	Precision	MSE
OrangeCounty, CA	9221	0.5335	0.0098
Seattle, WA	4930	0.5225	0.0163
Orlando, FL	6905	0.5356	0.0091

TABLE I
PRECISION AND MSE RESULT IN DIFFERENT CITIES; USING COSINE SIMILARITY METHOD

Cities	testing zones	Precision	MSE
OrangeCounty, CA	9221	0.536	0.0105
Seattle, WA	4930	0.5231	0.0178
Orlando, FL	6905	0.5363	0.0109

TABLE II
PRECISION AND MSE RESULT IN DIFFERENT CITIES; USING L2 NORM SIMILARITY METHOD

scheduling like us, we used a random strategy to assign values to each attribute of the schedule. We report the average precision and MSE of datasets corresponding to three regions to highlight the differences between our method and the baseline method in Figure 5. Our method outperforms the baseline in precision and MSE for each different K value.

*d) **Execution Latency**:* During the experiment, latency is also an important feature. To query one recommended schedule for a given zone, it takes **0.017ms**, **0.012ms**, and **0.014ms**, on Orange County CA, Seattle WA and Orlando FL dataset, respectfully. On average, the latency is typically small making the technique suitable for real-time application.

V. RELATED WORK

In this section, we briefly review related work on smart irrigation controllers and the recommended models used in the IoT area.

A. Smart Irrigation Controllers

To provide an optimized irrigation schedule, sensor-based smart irrigation controllers (e.g., [19]–[21]) typically use off-the-shelf components (e.g. Arduino boards, Raspberry-Pis) that can be programmed to deploy. Moisture sensors that

Cities	testing zones	Precision	MSE
OrangeCounty, CA	9221	0.5257	0.0312
Seattle, WA	4930	0.4565	0.0076
Orlando, FL	6905	0.5313	0.0085

TABLE III
PRECISION AND MSE RESULT IN DIFFERENT CITIES; USING INNER PRODUCT SIMILARITY METHOD

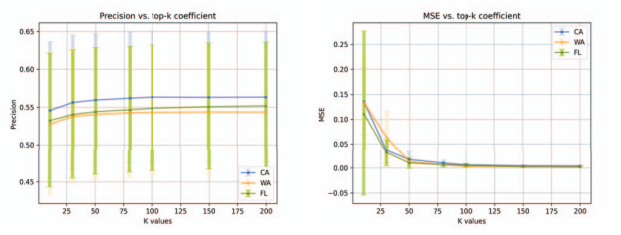

Fig. 4. Precision and MSE varies with top-k as parameter.

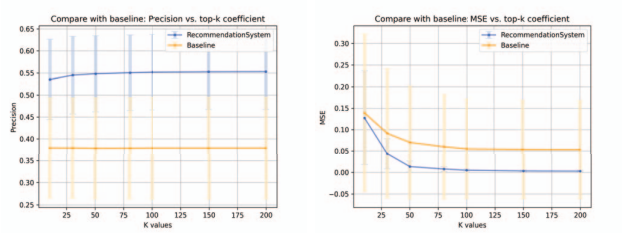

Fig. 5. Compare with baseline: Precision and MSE varies with top-k as parameter.

linked on those components can monitor the moisture level of soil. If the moisture level is below a threshold, the smart irrigation controller can trigger an irrigation event. Other than soil moisture sensors, [24] adds a sensor data analyzer to predict the irrigation schedule using truth tables that determine not just when to irrigate, but also where to irrigate. [16] utilizes PID-based control mechanisms that implement a feedback loop for precise irrigation control. Simulation-based approaches have studied the use of fuzzy methods, [26] ; such techniques are challenging to realize in commercial implementations on actual homeowner landscapes. Efforts to create sustainable solutions that consider not just effective water conservation but also aim to reduce overall energy costs have also been explored [25] A range of smart irrigation applications have been developed using ET-based irrigation controllers [27] - such devices measure ET and use water balance models in conjunction with other types of sensor data. However, since the ET value is closely related to local weather conditions, such apps are limited certain type of landscapes, crops and regions [28].

In the last decade, machine learning mechanisms such as regression models have been used for better irrigation scheduling. [29] considers irrigation planning as an optimal control problem that aims to minimize total irrigation usage. These algorithms leverage models derived whole-year weather data, that introduces uncertainty and complexity when applied to real deployments [28]. To estimate moisture data without sensors, [30], artificial neural network(ANN) and fuzzy logic approaches have been used to model and predict moisture data for irrigation scheduling. However, since the ANN models are trained using historical weather data, the prediction accuracy is impacted as the weather trends change,

B. Recommendation Model in IoT applications

A typical IoT ecosystem involves several devices that generate and upload heterogeneous data to cloud platforms, and utilize data analysis tools such as recommendation systems to mine and build different applications. For example, [33], [31] and [32] explore the use of recommendation systems for travel and tourism applications based on information from uploaded photos. From an algorithmic perspective, techniques used in traditional recommendation systems (RS) (such as movie /content recommendations on websites/web applications) such as collaborative filtering can also be used for recommendations

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 25,2024 at 07:37:24 UTC from IEEE Xplore. Restrictions apply.

in an IoT setting such as ours, the latter faces additional challenges due to cold start, diversity, and scalability [34].

VI. Conclusion

In this paper we present a recommendation system integrated in Rachio smart irrigation controller workflow. In absence of water usage data, we interpret water efficiency and user feedback through the irrigation event log. The Correlated Weighted Majority Vote is been designed as a recommendation strategy, offering user a better satisfaction over usage ratio watering plan. The experiments on three different cities dataset have shown that our recommendation system performs consistent over different area and weather conditions, along with a low execution latency that allows future real time deployment.

VII. Acknowledgements

This material is based on research sponsored by Rachio funding, and it is partially supported by NSF Grants No. 2420846, 2245372, 2133391, 2008993 and 1952247. The authors also acknowledge helpful feedback from Nada Lahjouji and Sriram Rao on earlier versions of this paper.

References

[1] Ekström, Joakim. "The phi-coefficient, the tetrachoric correlation coefficient, and the Pearson-Yule Debate." (2011).
[2] https://www.postgresql.org
[3] Zhu, X., Chikangaise, P., Shi, W., Chen, W.H. and Yuan, S., 2018. Review of intelligent sprinkler irrigation technologies for remote autonomous system. International Journal of Agricultural and Biological Engineering, 11(1).
[4] Yan H J. Study on water distribution irrigation uniformity of center pivot and later move irrigation system based on variable rate technology. Doctoral Thesis, China Agricultural University, 2005; pp.95–96.
[5] Peters T R, Evett S R. Automation of a center pivot using the temperature-time threshold method of irrigation scheduling. J Irrig Drain Eng ASCE, 2008; 134(1): 286–291.
[6] Dukes, M.D., 2020. Two decades of smart irrigation controllers in US landscape irrigation. Transactions of the ASABE, 63(5), pp.1593-1601.
[7] Carr, M.H. and Zwick, P.D., 2016. Florida 2070: Mapping Florida's future—Alternative patterns of development in 2070. Gainesville: University of Florida Geoplan Center.
[8] https://www.drought.gov/states/california
[9] https://water.ca.gov/Programs/Water-Use-And-Efficiency/Urban-Water-Use-Efficiency/Model-Water-Efficient-Landscape-Ordinance
[10] Zhang, X. and Khachatryan, H., 2019. Investigating homeowners' preferences for smart irrigation technology features.Water,11(10), p.1996.
[11] WaterSense product search, irrigation controllers. Washington, DC: U.S. Environmental Protection Agency. Retrieved from https://lookforwatersense.epa.gov/Product- Search-Results-IrrigationController.html
[12] Haley, M. B., Dukes, M. D., Miller, G. L. (2007). Residential irrigation water use in central Florida. J. Irrig. Drain Eng., 133(5), 427-434.
[13] Lunstad, N.T. and Sowby, R.B., 2024. Smart irrigation controllers in residential applications and the potential of integrated water distribution systems.Journal of Water Resources Planning and Management,150(1), p.03123002.
[14] Hayk Khachatryan, Dong Hee Suh, Wan Xu, Pilar Useche, Michael D. Dukes, Towards sustainable water management: Preferences and willingness to pay for smart landscape irrigation technologies, Land Use Policy, Volume 85, 2019, Pages 33-41, ISSN 0264-8377.
[15] Xiang, K., Li, Y., Horton, R., Feng, H. (2020). Similarity and difference of potential evapotranspiration and reference crop evapotranspiration – a review. Agricultural Water Management, 232, 106043.
[16] Jain, R.K., 2022, June. Modeling, Simulation, and Setting the Control Parameters for Automation of Irrigation System Using PID and ANN methods. In E-business technologies conference proceedings (Vol. 2, No. 1, pp. 48-56).
[17] Shruthi, G., Kumari, B.S., Rani, R.P. and Preyadharan, R., 2017, April. A-real time smart sprinkler irrigation control system. In 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE) (pp. 1-5). IEEE.
[18] Meyer, T.C. and Hancke, G.P., 2015, November. Design of a smart sprinkler system. In TENCON 2015-2015 IEEE Region 10 Conference (pp. 1-6). IEEE.
[19] 7. S. Koprda, Z. Balogh, D. Hruby and M. Turcam, "Proposal of the irrigation system using low-cost arduino system as part of a smart home", Proc. IEEE 13th Int. Symp. Intell. Syst. Inform., Sep 2015.
[20] J. Karpagam, I. I. Merlin, P. Bavithra and J. Kousalya, "Smart irrigation system using IoT", Proc. 6th Int. Conf. Adv. Comput. Commun. Syst., pp. 1292-1295, 2020.
[21] B. V. Raju, "An IoT based low cost agriculture field monitoring system", J. Appl. Sci. Comput., vol. 4, no. 4, pp. 128-136, Apr. 2019
[22] C. Stolojescu-Crisan, B. -P. Butunoi and C. Crisan, "An IoT Based Smart Irrigation System," in IEEE Consumer Electronics Magazine, vol. 11, no. 3, pp. 50-58, 1 May 2022.
[23] M. Mendez, J. Carrillo, O. Martin, C. Tchata, P. Sundaravadivel and J. Vasil, "EasyYard: An IoT-Based Smart Controller for a Connected Backyard," 2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), Rourkela, India, 2019, pp. 257-261.
[24] Masaba, K., Ntakirutimana, A. and Ustun, T.S., 2016. Design and implementation of a smart irrigation system for improved water-energy efficiency.
[25] Yin, Y. and Li, X., 2017. Research of Smart and Remote Sprinkler System for Lawn. In 2nd International Conference on Mechatronics and Information Technology. Francis Academic Press, UK (pp. 401-404).
[26] Aggarwal, P. and Thakur, A., 2019, March. Fuzzy Interface Automatic Brassica Horticulture Hoop House. In 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 297-302). IEEE.
[27] Migliaccio, K. W., K. T. Morgan, G. Vellidis, L. Zotarelli, C. Fraisse, B. A. Zurweller, J. H. Andreis, J. H. Crane, and D. L. Rowland. 2016. "Smartphone apps for irrigation scheduling." Trans. ASABE 59 (1): 291–301.
[28] Gu, Z., Qi, Z., Burghate, R., Yuan, S., Jiao, X. and Xu, J., 2020. Irrigation scheduling approaches and applications: A review. Journal of Irrigation and Drainage Engineering, 146(6), p.04020007.
[29] Lopes, S. O., F. A. Fontes, R. Pereira, M. de Pinho, and A. M. Gonçalves. 2016. "Optimal control applied to an irrigation planning problem." Math. Prob. Eng. 2016 (Jun): 1–10.
[30] Tsang, S., and C. Jim. 2016. "Applying artificial intelligence modeling to optimize green roof irrigation." Energy Build. 127 (Sep): 360–369.
[31] S. Cha, M. P. Ruiz, M. Wachowicz, L. H. Tran, H. Cao and I. Maduako, "The role of an IoT platform in the design of real-time recommender systems", Proc. IEEE 3rd World Forum Internet Things (WF-IoT), pp. 448-453, 2016.
[32] I. Mashal, O. Alsaryrah and T.-Y. Chung, "Testing and evaluating recommendation algorithms in Internet of Things", J. Ambient Intell. Hum. Comput., vol. 7, no. 6, pp. 889-900, 2016.
[33] Y. Zheng, X. Xie, Q. Yang and V. W. Zheng, "Collaborative location and activity recommendations with GPS history data", Proc. Int. Conf. World Wide Web, pp. 1029-1038, 2010.
[34] N. S. Nizamkari, "A graph-based trust-enhanced recommender system for service selection in IOT", Proc. Int. Conf. Inventive Syst. Control (ICISC), pp. 1-5, 2017.
[35] API design for machine learning software: experiences from the scikit-learn project, Buitinck et al., 2013
[36] Bruch, S., 2024. Foundations of Vector Retrieval. arXiv preprint arXiv:2401.09350.
[37] Pan, J.J., Wang, J. and Li, G., 2023. Survey of vector database management systems. arXiv preprint arXiv:2310.14021.
[38] Malkov, Y.A. and Yashunin, D.A., 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence, 42(4), pp.824-836.
[39] https://github.com/pgvector/pgvector
[40] Goyani, M. and Chaurasiya, N., 2020. A review of movie recommendation system: Limitations, Survey and Challenges. ELCVIA: electronic letters on computer vision and image analysis, 19(3), pp.0018-37.
[41] https://www.nytimes.com/wirecutter/reviews/best-smart-sprinkler-controller/

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 25,2024 at 07:37:24 UTC from IEEE Xplore. Restrictions apply.