HARVEST: a complete solution for smart agriculture monitoring

Ioannis Mavroudopoulos[1], Theodoros Toliopoulos[1], Anastasios Gounaris[1], Georgios Kynigopoulos[1], Andreas Andreadis[1] and Ilias Kalfas[2]

[1]*Aristotle University of Thessaloniki, Greece*
[2]*American Farm School, Thessaloniki, Greece*

Abstract
We present HARVEST, a complete end-to-end solution for smart agriculture monitoring. HARVEST is built using open-source systems specifically designed for big data applications to efficiently ingest and process real-world measurements collected in real-time from sensors located in different areas of Greece. The results from the analysis benefit not only the farmers but also additional roles in smart agriculture, such as data analysts and sensor infrastructure operators. HARVEST encapsulates advanced analytics to fill missing measurements, detect malfunctioning sensors and produce meaningful insights to support decision-making thus departing from common data warehousing deployments.

1. Introduction

The agricultural industry nowadays aims transition to smart agriculture through the use of the Internet of Things (IoT) and big data technologies. Smart agriculture applications boost operational efficiency and productivity [1]. As shown in recent surveys, e.g., [1], smart agriculture applications are typically divided into 4 categories, namely monitoring, tracking and tracing, smart precision farming, and greenhouse production. We focus on the first category that has received the highest attention.Its core rationale is that important factors affecting farming and production, such as soil temperature and air humidity, need to be measured by IoT sensors and transferred into a (typically) cloud-based infrastructure. Then, using machine learning (ML) and other big data analytics techniques, useful information and insights are mined. The results of the analysis provide recommendations to the farmers about their farming and irrigation plans, which result in optimized production while at the same time minimizing labour costs.

Some of the monitoring applications do not use a central storage. They rather collect, process and visualize data to a device that is connected with the sensors in the field. In these applications the processing takes place either on an edge device (like Raspberry Pi) or on a commodity machine, e.g., [2], which poses limitations due to lack of available compute resources. On the contrary, applications that move data from the sensors to the cloud enable more advanced analytics and benefit from efficient data storage [3]. In several cases, even though the data are collected in the cloud, they are processed separately for each user, which creates limitations regarding the employed analytics methods and is difficult to compensate for the high costs to set up and manage a large-scale infrastructure. Therefore, taking advantage of all the available data to more efficiently support the end-users' decision making is the most attractive option. Data from custom sensors are pushed to the cloud, where they are processed collectively using ML and AI to learn and understand the behavior of different crops in different parts of the world. However, HARVEST, which is our proposal, goes one step further, as it enables different roles in smart agriculture, like farmers, data analysts, and network operators, to interact and benefit from this large-scale monitoring application.

In this work, we introduce HARVEST, a complete solution for smart agriculture monitoring that effectively supports big data analytics. It is complete in the sense that it is an end-to-end system and serves the needs of all main stakeholders: farmers, sensor infrastructure operators and data analysts. In our scenario, we collect data from the LoRa IoT Network of the American Farm School (AFS) of Thessaloniki, Greece, which produces over 70K measurements per day and covers more than 1/4 of the total area of Greece. HARVEST can also detect outliers, fill missing values and extract useful insights, which is typically not included in data warehousing deployments. To achieve its objectives, HARVEST aims to fulfil the following functional requirements below: **R1:** Efficiently ingest and process big data, ensuring low latency and scalability. **R2:** Inform infrastructure operators for anomalies and malfunctioning sensors. **R3:** Provide easily understandable analytical results and visualizations for the farmers to assist them in the day-to-day activities. **R4:** Equip data

Proceedings of the 6th International Workshop on Big Data Visual Exploration and Analytics co-located with EDBT/ICDT 2023 Joint Conference (March 28-31, 2023), Ioannina, GR
mavroudo@csd.auth.gr (I. Mavroudopoulos); tatoliop@csd.auth.gr (T. Toliopoulos); gounaria@csd.auth.gr (A. Gounaris); kynigopo@csd.auth.gr (G. Kynigopoulos); andreadis.dws@gmail.com (A. Andreadis); ikalfa@afs.edu.gr (I. Kalfas)

analysts with built-in functionalities for filling missing values and automated extraction of insights to facilitate data exploration.

2. HARVEST Infrastructure

The data used are collected in real time from the American Farm School (AFS) of Thessaloniki's LoRa IoT Network, which operates in many areas of Greece. The most important components of the network are the nodes and the gateways. Nodes are specifically designed to operate in open environments, like fields and farms, and each node contains a variety of sensors. These sensors provide real-time measurements of environmental variables, such as humidity and temperature. In addition to the nodes, the AFS network also includes meteorological stations, which can capture additional measurements like solar radiation. At the time of the study, the AFS LoRa network had 254 nodes and 50 meteorological stations, with an average number of measurements per day of 70K.

Gateways operate as routers between the nodes and the cloud, where the data are stored. The nodes utilize LoRaWAN[4], a low-power network, to send packages to the gateways, while the gateways employ a high-bandwidth network, i.e. WiFi, to transfer these packages to the cloud. LoRaWAN allows a gateway to receive packages from nodes that are installed in up to a 25-kilometer distance. Therefore, with only 62 gateways, the network coverage is estimated to be 36.6 million acres, which is more than 27% of the total area of Greece.

2.1. System Overview

Fig. 1 depicts a high-level overview of the proposed infrastructure, starting from the sensors and ending with the analytics. In this section, we briefly discuss the various components of our solution.

LoRa Network sensors. The network consists of 13 different types of nodes, each with a unique collection of sensors. The most common type of node is the one installed in the fields, which accounts for about 68% of the total nodes. The sensors embedded in this type of node measure the air pressure, humidity, and temperature, as well as the dew point, the soil temperature, and the volumetric water content. All the measurements are transmitted from the sensors to the cloud through the gateways in order to be stored. Then, they become available through an API.

Data Collector. This module is implemented using a simple Java software and is responsible for periodically retrieving new measurements from the AFS network API. Since the measurements come from different node types, a pre-processing step is required to standardize their structure. Moreover, additional fields are added to each

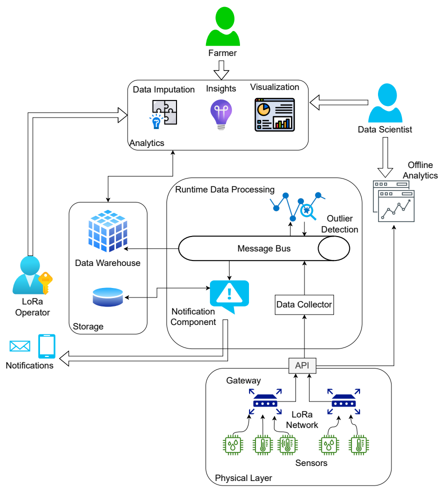

Figure 1: System overview

measurement, such as the district name in which the sensor operates, in order to provide more meaningful spatial indexing later. The final measurements are ingested to the Message Bus.

Message Bus. This component enables information flow between the various components in HARVEST. Each component of the Runtime Data Processing component pulls data from the Message Bus, processes them, and then writes them back to a different channel. In our implementation, we use Apache Kafka, an open-source distributed event streaming platform, widely used in the industry sector for high-performance data pipelines, streaming analytics, and data integration **(R1)**.

Outlier Detection. Some of the data ingested by the Data Collector might have errors or inconsistencies, which, if left unattended, would have a negative impact on any produced result. The Outlier Detection component is implemented in Apache Flink and is responsible for eliminating anomalies before the permanent storage of the data in the Data Warehouse. We have implemented two different approaches to anomaly detection. The first one is a simple rule-based technique that detects anomalous behaviors, like out of bounds values or inactive nodes, with the rules provided by domain-experts. The other method uses a simple continuous outlier detection technique to detect measurements that significantly deviate from previous measurements of the same node or from measurements made by similar sensors in other nodes. This unsupervised technique allows HARVEST to eliminate more complex anomalies that would have been missed by the naive rule-based method. All the detected anomalies are removed from the main stream and written into a different topic of the Message Bus in order to be handled by the Notification Component.

Data Warehouse. After removing the outlying measurements, the data are forwarded to the Data Warehouse (DW) via the Message Bus. DW is responsible for efficiently storing the multi-dimensional measurements and enabling Online Analytical Processing (OLAP) operations. Additionally, the stored data are then utilized by the methods in the Analytics layer. In our implementation, we have employed Apache Druid, an open-source real-time database that automatically integrates with Apache Kafka and provides fast and consistent queries at high concurrency **(R1)**. For each measurement, except for the value and the timestamp, we maintain the sensor id, node id, node type, minimum and maximum values that are acceptable for this particular sensor type, and spatial information (longitude, latitude, municipality, prefecture, and district). This information allows aggregations at various levels.

Notification Component. It is crucial for the AFS LoRa operators to get notifications of any anomalous behavior. This allows them to detect and fix malfunctioning nodes and sensors or detect possible natural disasters, like fires or floods. The Notification Component's responsibility is to inform the network operators of any anomalies discovered by the Outlier Detection component **(R2)**. This component is implemented in Python. The incidents are saved in a relational database to avoid sending alerts for the same incident more than once, and notifications (in the form of emails and Viber messages) are sent periodically.

Analytics. A number of different analytics are applied on top of the ingested data. Also, a visualization tool that is easily configurable to match each user's unique demands has been employed **(R3)**. In our implementation, we have used Grafana, an open source tool for creating custom dashboards and that is easily integrated with both DW and the database that stores the outliers. Next, we have the Offline Analytics tool implemented in Python, which extracts statistics (e.g., active nodes and average measurements per day) about the LoRa network in order to support future decision-making concerning the infrastructure **(R2,R4)**. Finally, on top of the DW, we have implemented two advanced analytics methods, top-k insights extraction and data imputation. We discuss these methods in the following sections. Note that the structure of the proposed infrastructure enables the implementation of various other analytical methods, like the prediction of a sensor failure, without interfering with the rest of the system. In other words, the system is modular and extensible by design.

2.2. Data Imputation and Insights Extraction

Data transmitted over the network may contain missing values for various reasons. Lost packages due to communication errors between the node and the gateway is a very common phenomenon. Additionally, hardware damage and sensor malfunction, which can be caused by extreme weather conditions, also produce large missing blocks of data. Therefore, even under continuous monitoring, the sensors often remain inactive for several hours, or even for days, until the engineer fixes the detected damage. In HARVEST, we employ data imputation techniques to fill in the missing values. This component reads and writes directly to the DW and is implemented using Python. We have implemented and evaluated the performance of a number of different methods, e.g., methods that utilize the temporal dependencies and correlations between different sensors in the same node to better approximate the missing values and methods that take advantage of the structure of the network, utilizing the correlations between the same variables in multiple nodes to enhance the accuracy of their predictions. Deep learning techniques are more accurate in predicting the missing measurements when operating under the multi-node setting. However, since data imputation is a plug-in to the main system, in the future, we can easily test additional techniques and re-evaluate our choice.

Up to this point, we have discussed the data ingestion along with preprocessing steps, like outlier detection and data imputation, in order to enhance the quality of the data. However, another important role of HARVEST is to extract knowledge from the data to support the end-users in their daily activities. OLAP, enabled by the DW, can be used to obtain useful information about the stored data, but it takes a lot of effort to manually pose queries and interpret the results, particularly for non-expert users like farmers. Therefore, we need an automated way to obtain insights, where insights are interesting observations that derive from aggregated data. Additionally, there should be a way to rank the various insights with an appropriate score function so that only the most interesting ones are kept. The extracted information will provide informative summaries of the data to non-expert users while also aiding data analysts in data exploration. To satisfy this requirement, we have integrated and extended the method described in [5] into our system. This method extends OLAP tools' simple aggregations by conducting analysis operations on them (e.g., rank, difference), resulting in the extraction of valuable information. This component requires the following input parameters. *Participating attributes*: The user can define a subset of the available attributes (e.g., the timestamp of the measurements and the region of the sensor that produced them) in order to generate more meaningful insights. *Measure column*: The column on which aggregations will be conducted on (e.g., temperature values). k: The number of top insights that will be extracted As the value of the parameter k increases, the process time to calculate the top-k insights increases; typically, a value between 50

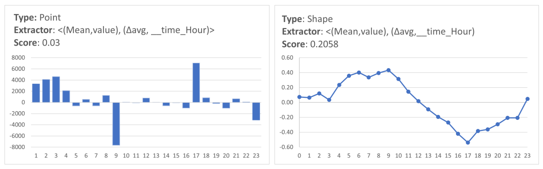

Figure 2: Left: average air pressure difference from the previous hour (point insight). Right: average temperature difference through-out the day (shape insight).

and 100 yields good results. *Aggregator*: The aggregation function, which could be either SUM, COUNT or MEAN. *Extractor*: Currently, in our system we support 4 types of extractor: (1) *Previous Difference*, which subtracts the current value from the previous one (this applies to attributes that can be sorted, e.g. years), (2) *Rank*, which simply ranks the values, (3) *Percentage*, which finds the percentage of each attribute and finally (4) *Average Difference*, which subtracts the average aggregated value from the initial one. τ*-depth*: The number of extractors that will be used. A value of 1 indicates that only the aggregator will be used. The recommended value is 2 (aggregator + 1 extractor) as more extractors will raise the complexity exponentially without providing better results. *Insight types*: HARVEST supports 5 insight types: (1) *Point* insights mark an outstanding outlier, which is far greater from the other set, (2) *Shape* insights show a trend that is increasing or decreasing rapidly, (3) *Attribution* that shows outstanding percentage of an outlier, (4) *Two Points*, which reflects the difference of two points from the others and finally (5) *Last Point* which refers to an outstanding outlier, which is very small compared to the set [6]. Two examples are given in Fig. 2.

We built a user-friendly interface so that any actor can log in and quickly run a top-k insights query.[1] Insights, similar to the Data Imputation component, is a plug-in to the core system, i.e., it connects directly with the DW and does not interfere with any other component, making it easy to extend or modify. This component is implemented in Javascript and is publicly available.[2]

3. Deployment and Demonstration

The system has been running for over 20 months on a Linux server with 16GB RAM, a 2.3GHz CPU with 8 cores, and 200GB of hard disk space. During that period, a total of 22 million measurements have been successfully ingested and stored. All the services were containerized to minimize the installation process and increase interoperability. That is, if the AFS LoRa network expands in the future, the proposed system can be easily transferred to a new environment that can match the new demands.

The demonstration will present how the data are extracted from the network's API in real-time, processed by the various components before stored in the data warehouse. Additionally, we will show how the different actors can benefit from HARVEST. **End-users**, e.g., farmers, can utilize the custom graphs in the Visualization component to monitor the crops in real-time and make better day-to-day decisions, such as deciding when to water the plants according to the measured soil humidity. Additionally, they can use the Insights UI to obtain interesting insights about the collected data without spending any time configuring the system and/or submitting queries . **Data Analysts** can also benefit from the proposed framework in multiple ways. First, insights can be extracted from both the Offline analytics and Insights components, in order to provide some initial information about the data and the network structure. Next, using the OLAP tool provided by the DW, the analyst can efficiently submit SQL queries over the multi-dimensional data. Finally, the HARVEST's structure enables the implementation and evaluation of additional components, like CEP-based smart agriculture and predicting the remaining useful lifetime of a sensor. We will demonstrate how such extensions can be realized. **Operators of the LoRa Infrastructure** can continuously monitor the LoRa network in the Visualization component while also receiving real-time notifications for possible malfunctions.

[1]https://kinigopoulos.github.io/top-k-insights/
[2]https://github.com/Kinigopoulos/top-k-insights

References

[1] V. K. Quy, N. V. Hau, D. V. Anh, N. M. Quy, N. T. Ban, S. Lanza, G. Randazzo, A. Muzirafuti, Iot-enabled smart agriculture: Architecture, applications, and challenges, Applied Sciences 12 (2022) 3396.

[2] G. Lee, M. Kim, K. Koroki, A. Ishimoto, S. H. Sakamoto, S. Ieiri, Wireless ic tag based monitoring system for individual pigs in pig farm, in: 1st Global Conf. on Life Sciences and Technologies (LifeTech), IEEE, 2019, pp. 168–170.

[3] W. Zou, W. Jing, G. Chen, Y. Lu, H. Song, A survey of big data analytics for smart forestry, IEEE Access 7 (2019) 46621–46636.

[4] S. Ali, T. Glass, B. Parr, J. Potgieter, F. Alam, Low cost sensor with iot lorawan connectivity and machine learning-based calibration for air pollution monitoring, IEEE Transactions on Instrumentation and Measurement 70 (2020) 1–11.

[5] B. Tang, S. Han, M. L. Yiu, R. Ding, D. Zhang, Extracting top-k insights from multi-dimensional data, in: SIGMOD, 2017, p. 1509–1524.

[6] R. Ding, S. Han, Y. Xu, H. Zhang, D. Zhang, Quickinsights: Quick and automatic discovery of insights from multi-dimensional data, in: SIGMOD, 2019, p. 317–332.