A Lambda Architecture for Imbalance Prediction with Smart Cane

Oussema Fakhfakh, Imen Megdiche, Réjane Dalcé, Thierry Val

A Lambda Architecture for Imbalance Prediction with Smart Cane

Oussema Fakhfakh[1], Imen Megdiche[1,2], Réjane Dalcé[1,2], Thierry Val[1,3]
[1]Institut de Recherche en Informatique de Toulouse, France
[2]Institut National Universitaire Jean François Champollion, ISIS Castres, France
[3]Université Toulouse 2 Jean Jaurès, France
oussema.fakhfakh@irit.fr , imen.megdiche@irit.fr, rejane.dalce@irit.fr, thierry.val@irit.fr

Abstract - Providing medical staff with a system for imbalance prediction can be a way to track frailty and to monitor patient's state more accurately, since imbalance is assessed as a warning sign of frailty. The purpose of this paper is to propose a complete architecture able to track imbalance from IoT devices, store their data in an adapted environment and process them with advanced analytics. Our solution consists of a Lambda Architecture that collects data from a smart Cane used by elderly people and processes in real-time and batch ways these data to predict imbalance.

Keywords: Lambda Architecture, Deep Learning, Imbalance Prediction, frailty.

I. INTRODUCTION

With the technological advancement and the emergence of Internet of Medical Things (IoMT) tools, several solutions based on these concepts have been developed in order to tackle the problem of frailty detection. While existing solutions focused on fall detection, this approach is unfortunately not ideal when it comes to frailty detection: to trigger the system, the elderly must fall which may lead to injuries, both physical and psychological. In a worst-case scenario, the fall possibly leads to death. It is therefore crucial to intervene earlier in order to minimize damage, by focusing on symptoms that may indicate the onset of frailty, for instance imbalance. The study in [1] indicated that imbalance can be considered a warning sign of frailty.

This paper introduces a complete architecture that covers the gathering of data from an IoMT device, its storage and processing. This architecture aims to generate indicators of frailty for the elderly users of the IoMT device. Our proposal is of lambda architecture type covering the requirements of analyzing both in real time and in batch big data of IOT. This kind of architecture has proved its efficiency for real time and batch analysis in other context such as Smart Grids [3].

The paper is structured in three sections. In section II we present our lambda architecture. In section III we present a brief picture on the exploitation of data and we conclude in section IV.

II. A LAMBDA ARCHITECTURE FOR IMBALANCE PREDICTION

A. *Architecture overview*

Our proposed architecture complies with the concept of

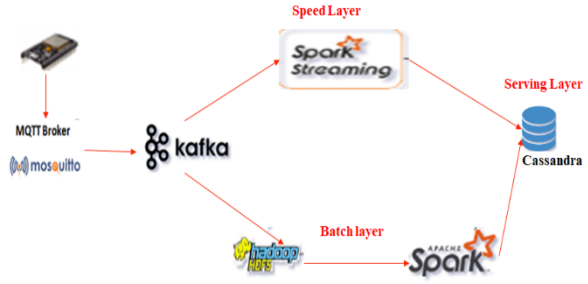

Figure 1. A Lambda Architecture for Imbalance Prediction

Lambda Architectures. A lambda architecture is a data storage and processing to handle massive quantities of veloce data that takes advantages of both batch and stream-processing techniques. For the purpose of imbalance prediction, our proposed architecture is able to store the massive IoT data coming from smart cane, which are used continuously by elderly people. These data are processed in real time by deep learning models which are able to predict if there is imbalance (but also fall) and serve this data to therapists. This architecture can handle data of several participants (data are stored according to GPDR requirements after collecting participants' consents). The massive quantity of data collected constitute a corpus for walking study.

In Figure 1, we depict an overview of our architecture composed of sensor data that are ingested by two layers : (i) *the speed layer* is able to treat in real time the data according to previously defined processes, in our case, we predict for each data if there is imbalance or not with a previously designed model, (ii) *the batch layer* will store data and process them in batch, in our case we fine-tune deep learning models. The serving layer is dedicated for

data consumption by final users, in our case the doctors can access dashboards of our processed data.

In the following sections, we explain in detail all the components of this architecture.

B. *Sources of ingested data*

In order to study the imbalance, our lambda architecture should collect labelled data to be able to train machine learning models and produce intelligent models for imbalance prediction. The ingestion is namely composed of raw motion data generated by a smart stick and a labelling mechanism. The labeling process is ensured by a web application that can be used when observing walking people. In the following, we describe both components: (i) the smart stick which extends the work of [2] and (ii) the labeling application.

1) *Smart Stick*

Our smart stick is presented in Figure 2. At the bottom of the stick we find an embedded system based an ESP32 board. It contains a 3D Accelerometer measuring linear acceleration and a 3D magnetometer capturing the variation of the magnetic field. We have then nine raw features collected from the stick: (lx,ly,lz) for acceleration, (mx,my, mz) for magnitude and (raw, pitch , yaw). Our embedded system is presented in Figure 3. The ESP32 is a single chip combo that supports short-range wireless communication namely WiFi and BLE. The cane also supports LoraWAN communication technology thanks to the integration of an RFM95 chip and the LMIC library.

Figure 2. The Smart Cane prototype

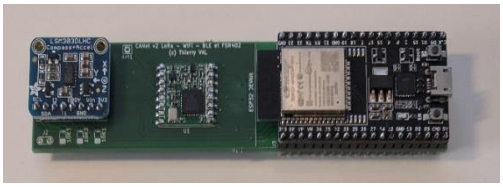

Figure 3. The embedded system

2) *Labeling Web Application*

We developed a web application named “Walkane”, as shown in Figure 4, that serves for the observers of the walking experimentations. For retirement homes, the therapists can use this application while running the Tinetti tests or, more generally, when supporting elderly people. The “Walkane” application can generate several experimentation campaigns. Based on the environment in which the experience will take place, the user configures through the application the list of characteristics of the environments such as doors, slopes, stairs, falls and so on. During the walk experimentation, the observer will tag the various events and label manually if the imbalance occurs. The output of the Walkane application completes the cane data with a feature for the environment and the binary label of imbalance (0/1) which is the target of the prediction. The synchronization between both data is set up using the BLE connection supported by the ESP32 board. Concretely, once the observer selects the cane of the observed user, a message is sent via the BLE interface to ask the ESP32 to start the measurements and the data recording from the sensors integrated therein. When the experiment ends, the same mechanism is repeated with another value of the message to stop recording.

Figure 4. Configuration of the Walkane Application

C. *Storage and processing modules*

In our lambda architecture, we combined different modules:

- **MQTT Broker**: Data sources of the big data architecture receiving records from the sensors mounted in the smart cane.
- **Kafka**: Messaging system linking the data sources and the Speed layer.
- **Spark Streaming**: This component illustrates the speed layer. Spark streaming allows data to be processed in real time and then to be stored in HDFS (Hadoop storage system) and Cassandra (NoSql Store). It is also able to perform predictions in real-time records using a trained deep learning model and to store the prediction results in Cassandra.
- **HDFS**: This file system is responsible for persisting the sensor data arriving in real time from Spark Streaming in order to build historical

records. It is also responsible for retrieving annotation files from the web application.

- **Spark**: Spark Engine is responsible for performing the batch processing. The batch processing consists of carrying out two tasks :(i) Merging sensor's data and annotation's data in order to retrieve the final set of data, (ii) Training a deep learning model on historical data and evaluating it in order to inject it into Spark Streaming for real-time prediction. Evaluation results will be stored in Cassandra as well.

- **Cassandra**: It is the database responsible for storing the data received in real time and for serving the prediction results to the final users via a dashboard for example.

In the following, we explain the interconnections between the different modules.

Table 1. Configuration of the Kafka connector

Configuration item	Definition
name	Connector name
connector.class	This item is responsible for specifying that Kafka will be connected to MQTT. In our case, the connector class is set to MqttSourceConnector
tasks.max	The maximum number of tasks performed by this connector
kafka.topic	The kafka topic which will receive data from the broker MQTT
mqtt.client_id	Id of Kafka client
mqtt.clean_session	Boolean value: either the application will still receive data on connection in case of MQTT broker disconnection or no
mqtt.connection_timeout	The time interval after which the connection fails if kafka or MQTT Broker is not functional
mqtt.keep_alive_interval	A time interval measured in seconds If during this period no data flows over an open connection the MQTT client will generate a **PINGREQ** (connectivity request) to confirm that the connection is open and working
mqtt.server_uris	The address of the Mqtt broker, in our case it is **localhost: 1883** corresponding to mosquitto
mqtt.topic	The MQTT topic in which Kafka must register as a consumer or a client

1) Cane – MQTT Broker

As mentioned earlier, the MQTT broker is in charge of receiving the data from the cane. We have chosen the popular **Mosquitto** as our broker. Since the cane supports both LoRa and WiFi links, we have chosen to report the data over the WiFi in order to evaluate the pipeline. A more suitable approach based on LoRa is currently under study but is outside the scope of the paper. The chosen approach relies on the cane being a 802.11 STA, and communicating with a server socket on a computer. This server socket then publishes the data on Mosquitto. We take advantage of the presence of a BLE interface on the cane to control the experiment.

2) MQTT Broker – KAFKA

Once the data is available at the broker, Kafka is in charge of collecting it from the relevant topic and storing it in a temporary storage (Retention disk). Being based on the Pub/Sub concept, Kafka is well suited for this role as it makes both data recuperation and injection into the Big Data Architecture easier. We exploited the Apache Kafka Connect connector with the configuration given in Table 1. Kafka considers the data as a stream of bytes: processing them as sensor records will be done by Spark Streaming.

3) KAFKA – Spark Streaming

Once Kafka receives the data, it injects it into the speed layer for real-time processing and the Batch layer to perform advanced treatment.

In our case, Spark Streaming is the speed layer. Spark Streaming performs the format conversion in order to retrieve the measurements from bytes received by Kafka, makes real-time prediction on the data using a prediction model trained in the batch layer and stores the data alongside the predictions in Cassandra and HDFS.

4) Spark Streaming – HDFS

Spark Streaming uses the Hadoop Storage System HDFS to store the sensor data and the annotation files. HDFS has been chosen because Spark is part of the Hadoop ecosystem: we can therefore expect relatively easy integration.

In HDFS, data is organized in directories and various file formats are available. We selected the csv format for the sensor data. The annotation spreadsheets generated by the web application are asynchronously copied to the annotations directory in order to perform batch processing with the records coming from the cane.

5) HDFS – Spark

The HDFS-Spark connection performs the batch processing which begins by merging the Sensor data file and Annotations data file. This operation relies on the timestamp field present in the files: in both files, a sample is collected every 100ms. The file obtained after this concatenation is used to train the deep learning model that is used by Spark Streaming for real-time predictions. The model predicts the Imbalance column.

6) Spark Streaming – Cassandra

Cassandra is a NoSQL database responsible for storing prediction results following the real-time processing and the batch processing.

The received records by Spark Streaming, are pre-processed and organized in order to be stored in a Cassandra table. Regarding Cassandra, we had to create a keyspace, which is a database, containing several tables. Afterwards, we create a table called SensorData in Cassandra to populate data coming from sensors mounted in the cane.

Cassandra is also responsible for storing predictions made on real-time records. These instantaneous predictions are carried out thanks to the model trained on the batch layer (based on historical data persisted in HDFS) and injected into the Speed layer (consisting of Spark Streaming). This task is part of the future improvements to perform to our solution.

7) Spark – Cassandra

Spark reads data in batch mode from HDFS, pre-processes it, merges the sensor data and annotation information and finally carries out predictions. These predictions aim at predicting information that is useful to the medical staff. For now they are stored in a Cassandra table along with the F1_score, Accuracy and Loss are computed.

8) Spark – Spark Streaming

As we have mentioned above, Spark is also responsible for training data containing the sensor features as well as the imbalance label. Afterwards, this trained model will be injected to Spark Streaming after being saved in h5 format in order to perform real-time predictions. This interconnection was not set-up in the context of this project. It represents as well a potential improvement of our solution.

III. EXPERIMENTAL RESULTS

A. Experimental Scenarios

Due to the current pandemic COVID-19, we were not able to gain access to the partner nursing home, EHPAD AGIR, Castres, France. Instead, we had to recruit researchers and engineering students in order to perform the scenarios. Similarly, we will focus on short range tests using WiFi links since our ability to test outside was severely limited by the lock down. Four young adults participated in the realization of the tests listed in Table 2.

B. Deep Learning model and results

To experiment these data, we developed an extended version of the Artificial Neural Network proposed in [1]. The major differences brought are: (i) the number of neurons in the input layer that depends on the environment features selected by the user when filling in the campaign settings in the web application, (ii) among the hidden layers we placed a dropout layer in order to tackle the over-fitting problem, (iii) some hyper-parameter optimizations are added to the new version of the model.

This model was experimented on the dataset collected from the second scenario. The dataset was divided into 1710 records for training and 840 records for test (70%,30%). Despite the small amount of data, the achieved
accuracy on test set exceeds 94% while the test loss is evaluated to be almost 0.18. Our results are encouraging, they confirm the potential of deep learning techniques in predicting imbalance based on motion data.

Table 2. Experimental scenarios

Scenarios	Hurdles	Events	Venue
Scenario1	-Stairs -2 doors	-Normal walking - Go down-stairs - Go up-stairs	Hall upstairs
Scenario2	-Stairs -2 doors	-Normal walking - Go down-stairs - Go up-stairs - 2 imbalances	Hall upstairs
Scenario3	-Slope -2 doors	-Normal walking - Go down the slope - Go up the slope	Connected Health Laboratory

IV. CONCLUSION AND PERSPECTIVES

This paper has shown a big data architecture in the form of a data tunnel which aims to detect a possible imbalance in the person motion and therefore to inform the geriatrician or the doctor of a possible frailty state. After having being fed into the Big data architecture, the data will be injected into the Artificial Neural Network in order to predict the imbalance. This model will be used in the speed layer to perform real-time tracking of possible cases of frailty. It will also be used in the batch processing layer to update the predictions over the entire dataset. Finally, the prediction results will be linked to a web interface facilitating access to the results by end users. Among the improvements that we plan to carry out on our solution, we aim to complete the interconnection between Spark and Spark Streaming in order to perform real-time predictions.

ACKNOWLEDGMENT

We would like to thank Toulouse Tech Transfer that financed the CIPAD project.

REFERENCES

[1] O. Fakhfakh, I. Megdiche, R. Dalcé,, and T. Val.” Imbalance Prediction Among Elderly People Using Deep Learning”. In 17th IEEE International Conference on Computer Systems and Applications AICCSA 2020, HOPE, An-talya, Turkey.2020.

[2] A. Lachtar, T. Val, and A. Kachouri. Elderly monitoring system in a smart city environment using LoRa and MQTT.IET Wireless SensorSystems, 2043-6386: 2019.

[3] A. A. Munshi and Y. A. I. Mohamed, "Data Lake Lambda Architecture for Smart Grids Big Data Analytics," in *IEEE Access*, vol. 6, pp. 40463-40471, 2018.