Ontology Modeling and Reasoning for Crowdsourcing Smartphone Testbed

Karan Shah and Carlos Rivero

Abstract—With the growing demand and usage of smartphones, it is estimated that by end of 2020 there will be more than a billion active devices. Recently smartphones are providing a new research trend in the field of crowdsourcing data generated by these smart phones. This data comes from different components of the device. For example the GPS sensor provides with location information of the user, the WiFi scanner finds access points nearby whereas the accelerometer sensor gives precise orientation about the device etc. This data can be collected and various data mining techniques can be applied to extract user pattern and behavior. Crowdsourcing projects such as the PhoneLab, a large smartphone testbed aggregates phone data from android devices which can be used infer insightful information. However the data is stored in unstructured format hence performing data mining technique is difficult. In this paper first using a framework called Jena (an ontology framework) part of which converts the raw data into ontology an infrastructure is developed. The infrastructure consist of following task a) pipeline to load the ontology data into Oracle 12c database using Jena Adapter b) perform Oracles Data mining capabilities on the ontology. Based on the results we evaluate and provide insightful information which was otherwise hidden.

I. INTRODUCTION

Although we are able to track user data the major problem is that it is not being used to a full extent. PhoneLab, a large crowdsourcing smartphone testbed provides subsidized rates for its participants and provides us with basic raw data. This data includes information about location, battery, wifi, etc but there is no correlation between them. Hence this data directly makes no sense and we need an infrastructure that will process this data and find hidden information among them. Since the data is in unstructured format it cannot be used directly and thus we first convert into structured format. Ontology modeling has been used to prototype context aware systems and has good reasoning capabilities.

In this paper we develop an ontology framework using Apache Jena that captures and convert unstructured data to find relationship between the Phonelab data. PhoneLab is an open source smartphone testbed. The goal of phone lab is to allow researchers perform smartphone experiments and collect large data about how people use their smartphones. Researchers are allowed to tweak the android system to add instrumentation. Instrumentation is used to study the android system on how it work for charging pattern, application transition patterns or file access pattern by adding logs. The phone lab instrumentation platform currently contains 8 tags with 25 actions in 5 different categories. Data collected is stored in JSON object format. This measurement data needs to be processed if study of correlation between information is desired. This paper improves on the previous work of applying Ontology modeling to capture the measurement data and its relationship, developing rules to describe the implicit relationships of data, and designing queries to facilitate access to context information. The goal is to develop a mobile framework for automatic identification of new context based on implicit rules as well as statistical reasoning. Through the development of such framework, we make three main contributions in this paper: (i) introduce Ontology-based context modeling to large scale crowd-sensing smartphone testbed, (ii) setup infrastructure to store the model using Oracle 12c, and (iii) support data mining capabilities to be performed on the model to extract user behavior.

The final goal of the project is to be able apply data mining techniques on unstructured data and infer correlation among the phone data. The first step in the process would be to have an robust infrastructure in place which will help convert JSON data to ontology and push them into Oracle 12c database which supports semantic graph storage. Next we will perform data mining on the graph data using one of many available tools that oracle supports like R, oracle data miner, etc. and find user patterns and behavior.

This paper is organized as follow. Section 2 presents the existing works that are related. This is followed by section 3 with descriptions on modeling of PhoneLab data and new concepts for deriving new context information. Section 4 discusses the use of Jena Adapter to load RDF data into Oracle 12c. Followed by section 5 which highlights the Oracle Data Mining techniques and examples. The paper concludes in Section 6 and discusses future work.

II. PREVIOUS WORK

The current existing framework PLOMAR[1] first converts the raw JSON data into ontology model to represent entities and their relation. The advantage of using ontology model is it enables defining rules that help find relationships between the different ontology model and later using SPARQL to find new context information and uncover hidden user pattern/behavior. Previously we had been using Protege API to convert JSON data into ontology. Protege is very OWL centric and one does not have to worry about RDF because the data is treated as set of axioms rather than a triple. It provides a simple GUI to create Ontologies. To find the relation between different ontology one has to write SPARQL queries. However there are no internal API to store the graph into a database like Oracle. Among many available RDF library Apache Jena is the most flexible as it cover all formats of RDF including providing API for creating Ontologies. It
is more stable and a huge online community. Apart from that
it solves the most important task of storing RDF in Oracle.
We use apache Jena to convert the JSON structure into RDF
(Ntriple) format.

III. ONTOLGY MODELLING OF PHONELAB DATA

In this section, we show an example of raw data like loca-
tion, power battery and power screen collected by Phonelab,
create the ontology that will represent this raw data using
available framework, and also describe how to setup and
model data mining technique that will help extract context
information.

A. Creating an Ontology

An ontology is a formal naming and definition of the types,
properties, and interrelationships of the entities. Mobile
sensor data has been recorded by PhoneLab which performs
experimental instrumentation or system modification to fetch
40Gb of user data. The raw data is in JSON format and
a sizeable sample data is available freely. The sample data
consists of five different category of data including location,
battery, wifi, power and network. A good example of the
instrumentation to describe ontology is given below which
is the location data in Json format

```
Location-Misc-PhoneLab
{
  "Action":"android.location.LOCATION_CHANGED",
  "Counter":2139,
  "LocFormat":1.0",
  "Location":
  {"ElapsedRealtimeNanos":54026639248,
   "HasSneak":false,
   "IsFromMockProvider":false,
   "HasBattery":false,
   "Accuracy":1834,
   "Latitude":42.9931695,
   "Longitude":78.4742969,
   "Speed":0,
   "Time":1427045651782,
   "HasAltitude":false,
   "HasAccuracy":false,
   "HasBattery":false,
   "HasSneak":false,
   "IsFromMockProvider":false,
   "Provider":"fused",
   "HasAccurate":true
}
```
B. Setting up Oracle 12c

Oracle 12c supports the storage of semantic data and Ontology by providing the Oracle Spatial and Semantic graph support. Today there has been an increasing market demand for graph data and graph databases. Stored data can be queried using ontology assisted query language like SPARQL. As part of Oracle Spatial 12c, an option for Oracle Database 12c Enterprise Edition, Oracle delivers an advanced semantic data management capability not found in any other commercial or open source triple store. With native support for RDF/RDFS/OWL/SKOS standards, this semantic data store enables application developers to benefit from an open, scalable, secure, integrated, efficient platform for RDF and OWL-based applications. These semantic database features include (among others): storing, inference and query of RDF/RDFS/OWL/SKOS data and ontologies.

Storing large amount of semantic data can be slow and not feasible using INSERT statements. Instead bulk loading is the most efficient way of load large quantity of data. Oracle provides a Jena Adapter that provides a java interface to connect to Oracle to perform batch loading.

In order to store RDF triples the following are the steps for setting up the Oracle Environment

- Create a pluggable database.
  Note: Oracle 12c has a new architecture i.e. multitenant architecture which consists the concept of Container and Pluggable database. Container Database (CDB) is a primary database that contains multiple plugged-in databases. Many operations can be performed at the container level to reduce management costs. A database is created as either a CDB or a non-CDB. Pluggable Database (PDB) contains A set of schemas, objects, and non-schema objects that can be plugged and unplugged from a container database. The PDB appears to OracleNet and end users as a database in and of itself but is actually managed within a container that may have many PDBs.

- Before installing Jena there are series of steps required to be followed. Start SQL*Plus. At the Enter username prompt, enter the following to log-in as a privileged user: sys/oracle as sysdba
- Enable the RDF semantic support
  @%ORACLE_HOME%/md/admin/catsem.sql

- Create a new PDB by using the seed
  CREATE PLUGGABLE DATABASE projowl2 ADMIN USER karan IDENTIFIED BY shah
  FILE_NAME_CONVERT = ('pdbseed','projowl2');

- Create the RDF-USER tablespace and the semantic network by running the following command
  CREATE TABLESPACE rdf_users DATAFILE 'rdf_users01.dbf'
  SIZE 128M REUSE AUTOEXTEND ON NEXT 64M
  MAXSIZE UNLIMITED SEGMENT SPACE MANAGEMENT AUTO;
  EXEC
  sem_apis.create_rdf_network('RDF_USERS');

- Create a user for the pluggable database
  CREATE USER rajesh IDENTIFIED BY shah
  DEFAULT TABLESPACE = rdf_users;
  ALTER USER rajesh QUOTA UNLIMITED ON rdf_users;
  Grant necessary privileges to the user.
  GRANT CONNECT, RESOURCE TO rajesh;

At this point we have successfully created a new PDB. A semantic model was created in the PDB containing our model. Using basic SQL statement we can start inserting RDF (in triple format) into the database.

C. Loading RDF into Oracle

Oracle provide three different ways to load RDF semantic data into the model. They are

- The fastest method to load RDF data is the bulk load method or by appending data into the semantic data store from a staging table. Each record data consists of a row containing the three components subject, predicate, and object of an RDF triple and optionally a named graph. Although this is used for loading large amounts of data; however, it cannot handle triples containing object values with more than 4000 bytes.
- Oracle provides a Java Client interface to load or append data from an N-Triple format file into the semantic data store. This approach is called the batch load method. This option is slower than bulk loading, but it handles triples containing object values with more than 4000 bytes.
- The simplest and easy way to load into the application table is using SQL INSERT statements that call the
SDO-RDF-TRIPLE-S constructor, which results in the constructing RDF triple, to be inserted into the semantic data store. This method is convenient for loading small amounts of data.

```java
String szJdbcURL = "jdbc:oracle:thin:@localhost:1521/projOwl";
String szUser = "rajesh";
String szPasswd = "shah";
String szModelName = "Location";

Oracle oracle = new Oracle(szJdbcURL, szUser, szPasswd);
GraphOracleSem graph = new GraphOracleSem(oracle, szModelName);

graph.getBulkUpdateHandler().prepareBulk(is, // input stream
                                            "", // base URI
                                            "N-TRIPLE", // data file type: can be RDF/XML, N-TRIPLE, etc.
                                            "rdf_users", // tablespace
                                            null, // flags
                                            null, // listener
                                            null // staging table name.
                                          );

graph.getBulkUpdateHandler().completeBulk(
                                            null, // flags for invoking
                                            SEM_APIS.bulk_load_from_staging_table
                                            null // staging table name
                                          );
```

To load thousands to hundreds of thousands of RDF/OWL data files into an Oracle database, we use the prepareBulk and completeBulk methods as shown in the code above.

The above code can load triples of a graph or model into an Oracle database in bulk loading style. The prepareBulk method bypasses the Jena in-memory graph or model and takes a direct input stream to an RDF data file, parses the data, and load the triples into an underlying staging table. If the staging table and an accompanying table for storing literals do not exist, they are created automatically.

The prepareBulk method can be invoked multiple times to load multiple data files into the same underlying staging table. Once all the data files are processed by the prepareBulk method, you can invoke completeBulk to load all the data into the semantic network.

### IV. Oracle Advanced Analytics

The Oracle Advanced Analytics component of Oracle allows us to perform predictive analytics and get actionable insights from the database, thus empowering data scientist and business analysts to extract knowledge from otherwise underutilized large datasets. The component comprises of data mining algorithms with R scripts using SQL and R language. It also provides a GUI ie. Oracle Data Miner which is an extension in SQL Developer. In this project we will explore Oracle Data Miner and apply its data mining capabilities on RDF data.

A. Setting up Oracle Data Miner in SQL Developer

The oracle data miner comes pre installed within the SQL Developer. Note: Install SQL Developer 4.1 separately. Following are the steps to start using data miner

- Open the SQL Developer and connect to our pluggable database using a privileged user account. At this stage, you have created a database account for the data miner user, and have created a SQL Developer connection for that user.

- Before we install the Data Miner Repository in the database using an automated process, we first add tables into our PDB that we will use in our data mining workflow.

- Create a new project in our repository and further create a new workflow within the project. At this stage we have successfully installed ODM and can start developing data models.

On successful connection you will be able to see tables, views ,etc and also run queries through worksheet. Before we install the Data Miner Repository in the database using an automated process, we first add tables into our PDB that we will use in the our data mining workflow

- Add new connection and select the PDB. A message tells you that the Data Miner Repository is not installed in the database, and asks you if you want to install the Repository. Click Yes. This will download the data miner repository and you can now start creating workflow.

- Create a new project in our repository and further create a new workflow within the project. At this stage we have successfully installed ODM and can start developing data models.
B. Data Mining Techniques

Oracle Data Miner provides a broad range of data mining techniques and different algorithms to solve big data problems.

1) Classification: Classification is a data mining technique that assigns each individual record to a target category or a class. Most commonly used technique for predicting a specific outcome such as response or no-response, high or medium or low-value customer, likely to buy or not buy. Oracle Data miner provides different classification algorithms like Logistic Regression, Naive Bayes, Support Vector Machine and Decision Tree. It is possible to run one or more algorithm within a single workflow and compare results. In our example we will consider the insurance dataset and predict whether the customer is likely to buy or not buy the insurance package based on several attributes.

We construct a workflow that consists of many connected nodes which describe the data mining model. We perform the following steps to build our classification model which will predict customers who will buy insurance:

- Add a data source. Drag and drop the data source node from the right editor and select the table/view (insurance dataset). Select the attributes that would be considered during the classification and load them.

- Next step is to build the classification model. We select the classification model from the right editor. Connecting it to the data source and we get to the settings page where we can select the attribute we want to predict. We select target at buy-insurance and case ID as customerID. You can also select the different algorithms that you would want the model to run. We split the data into 30(train)/70(test) and run the model.

We will compare the results from all 4 algorithms to compare and views its results.

- On the right editor you will see a green check on all algorithm after successfully training the model. We now compare the results of each algorithm to see which one did the best. From the performance matrix and the lift measure we see that the Decision Tree model worked best and predicted with the best accuracy. The Generalized Linear Model also provided good accuracy. We further explore the Decision Tree model now.

We see the model and the different decision rules for every node which the model has calculated.

- To view the results we apply the model to make predictions on the data. We disable all classification algorithm except decision tree. Add a data node to store the results.

- We run the apply node and view the data in the output we can see that the record have been classified.
2) Clustering: Clustering is an unsupervised mining technique that groups related data together which are more similar to each other and less similar to other data in different groups. Oracle 12c has three different types of clustering algorithms available. K mean which supports text mining, hierarchical clustering, distance based. Orthogonal Partitioning Clustering which includes hierarchical clustering, density based. Expectation Maximization which performs well in mixed data problems (dense and sparse). We will perform the clustering techniques on the same insurance dataset. We perform the following steps to build our clustering model:

- Load the data source and explore the data.
- Before we start building our model, we split the data into two sections. The first set will be used to build the model while the second set will be used as testing or to apply the model. First, we create a sample build node from the transformation tab and assign 60 percent of data to it, and then we create another sample apply node and assign 40 percent data to it.
- Next, we add the clustering node in our workflow. We can change the number of cluster in the setting for each algorithm. We run the clustering model and compare the different results.
- We select the K mean model and you can see the different clusters formed by Oracle. They cannot be changed and our decision should be based on how we use them. Each cluster will have Centroid information and a set of rules.

- We can compare two clusters and extract information from them. We can compare two clusters and closely study the attributes to understand what the cluster represents. From the information below, we can clearly see that cluster 11 has customers with medium and long term value, and cluster 18 has customers with short or low term value.

3) Association: Association is a data mining technique that discovers the probability of the co-occurrence of items in the dataset. For example, if people who buy milk and eggs also buy bread, then the confidence that you also buy bread along with milk and eggs is high. Oracle data miner uses the Apriori algorithm to find the frequent item set. The frequent item set determined are used to develop the association rules. The data set on which we perform the
association is a dummy dataset. Each record is a transaction sample of purchases made at a grocery store.

We now create our workflow:

- We place the data source node and load the dataset. Select all the attributes. Next optional step as before is to explore the dataset.

- Select the Association model from under the Component Palette. We join the data source node to it. We edit the association node which only runs the apriori algorithm. The transaction Id are the attributes that uniquely identify each transaction which in our case is the transactionId. This enable grouping of all related transaction. The ItemId is the attribute of the we want to analyze. In our case let analyze the first item that the customer buys. Value is used to specify another column with transaction data to combine with Item id. We select existence to check if there is any type of common bundling among all the values of item ID.

- We can view the model and see the results.

As we can see there are three concepts. First is support which is the proportion of transactions in the data set that contain the item. Confidence is the proportion of occurrence of antecedent that result in consequent. Lift is the strength of a rule over the random co-occurrence of the antecedent and the consequent. In our example the antecedent is pip fruit and vegetable while the consequent is the sausage with 71 percent confidence. This mean if customer buys fruit and vegetable there is 71 percent chance he will buy sausage.

- We can use these rules to create a model detail. The model details node is useful when we do unsupervised learning to extract rules.

- We can see the model attributes with the support and confidence. We can also see SQL that used to generate the rules. To use this rules we need to make them persistent by storing them a in a table node.
4) Regression: Regression is a data mining technique which is used to predict a time series data. Profits, sales, mortgage rates, house values and square footage and similar values can be predicted using regression. For example in our example we will predict the number of bikes that will be rented based on time, weather, wind speed, weekend and other factors. The regression technique needs a training dataset with the target values known. In our example only after we have trained a dataset to observe the number of bikes rent can we predict the count. The training model estimates the value of target as a function of predictors for each case in build data. The relationship between predictors and target are summarized in a model which can be applied to the testing dataset in which we predict the target value.

Oracle data miner supports two algorithm for regression. The first is the multiple regression which is a classic statistical technique supporting text and transactional data. The second is the support vector machine supports text and wide data. We divide the data into two section the build sample (70 percent) and the build apply data (30 percent) and like previous methods we can explore the data. We connect the build sample to the regression model and select the target which is the CNT i.e. the count of bikes to be rented.

We can compare the regression model and see the confidence for each model. We see the multiple regression model performance to be much better than the SVM.

To avoid over fitting it is advised to use large training set. We have successfully trained our model and in the next we test the model against sample test data. We connect the build apply data to the apply node and run the model.

The model is tested by applying the above model and compare the target value with the known value.

The Root Mean Squared Error and the Mean Absolute Error are commonly used statistics for evaluating the overall quality of a regression model. Different statistics may also be available depending on the regression methods used by the algorithm.

5) Feature Selection and Extraction: Noise can reduce the effect of data mining. Noise can be missing data, invalid data or valid but irrelevant data. Some extra data columns may not be meaningful and not needed for building and test a mining model. For example suppose in the above of renting bike where we predict the count of bike that will rented per day or per hour. If we have information like the type of car the person has little or effect.

Sometime we columns maybe correlated and measure the same thing. For example weather and forecast can relate to the same thing and hence can skew the logic of the algorithm and affect the accuracy. Sometime dataset might contain may attributes which are the dimensions of processing space required by the algorithm. The higher the dimensionality of the processing space the higher the computation cost. Feature selection and extraction are two approaches to dimension reduction a] feature selection - selecting the most relevant attributes b] feature extraction - combining attributes into a new reduced set of features.

Next steps involves running the feature node and evaluating the results. From the results we can see the weather attribute is the most important attribute that determines the renting of bike. If any of the attribute have negative value then it is noise. Attribute which are zero or less make no contribution to the prediction and should be removed.
Feature extraction process transform the attributes into linear combination of original attributes. It is a reduction process unlike feature selection which ranks the attribute. It results in a much smaller and richer set of attributes. The model is more accurate because the attribute size i.e. dimensionality is smaller and attribute is more meaningful.

6) Anomaly Detection: Anomaly detection is a technique that would identify unusual cases within data which is seemingly homogeneous. The technique is mainly used to detect fraud, network intrusion and similar events that are important but difficult to find. In our example of bike rental we will try to find data for example if a bike was rented after shop hours or if a bike was never returned and stolen.

Anomaly detection is a form of classification where there is only one class in the training data. It predicts whether a data point is typical(1) or atypical(0) for a distribution. To achieve accuracy the model should be first trained on a dataset that consists of both typical and atypical examples so it can differentiate between them. Outliers are cases that fall outside this distribution. For example few days the median count of bike would be 100 and the mean count would be 125 but a few days may have significant higher count may be 300.

V. LEARNING USER BATTERY PATTERN

In this example we will create a model that will learn user power batter pattern. The first step in the developing the model would be to create table with attributes that would make a significant contribution.To query semantic data, use the SEM-MATCH table function. This function has the following attributes:

```sql
SEM_MATCH(
    query VARCHAR2,
    models SEM_MODELS,
    rulebases SEM_RULEBASES,
    aliases SEM_ALIASES,
    filter VARCHAR2,
    index_status VARCHAR2,
    options VARCHAR2,
    graphs SEM_GRAPHS,
    named_graphs SEM_GRAPHS
)
RETURN ANYDATASET;
```

There are two different ways to query triples from Oracle database SEM MATCH-based SQL statements and SPARQL queries using Jena. Even though the functionality of both query is similar they are different behaviorally. For instance lets consider two approaches

**Example 1**

```sql
select s, p, o
from table(sem_match('{?s ?p ?o}',
               sem_models('Test_Model'), ....))
```

**Example 2**

```sql
```

Both the query perform the similarly however, there are some differences. Query 1 (SEM MATCH-based) reads all triples out of the model. It does not differentiate among URI, bNode, plain literals, and typed literals. It does not unescape certain characters. While query 2 (SPARQL query executed through Jena) also reads all triples out of the model however, reads additional columns, to differentiate URI, bNodes, plain literals, typed literals, and long literals ensuring proper jena node object.

The power battery consists of different attributes like level, status, CurrentNow, plugType, etc which can be used to find the pattern of individuals. Since the data is represented as a triple it is not possible to directly combine multiple table attributes together. The solution to apply data mining to multiple tables would be to create a flat table using Views.
We create a simple view from the above attributes. The phonelab data consists of samples of 11 users for over a period of one month. Because the amount of data is small and unsuitable to accurately develop a model we go about exploring the possibility. We write a SPARQL query that will populate the table with the required attributes which will be the source to our data model. The view is a virtual view meaning the content in the view will change depending on the power battery table data.

```
FROM TABLE(?data)
WHERE {
  ?data a phonelab:PowerBattery.
  ?data phy:plugtype ?plugType.
  ?data phy:technology ?technology.
  ?data phy:level ?level.
  ?data phy:batteryLevel ?batteryLevel.
}
```

![Fig. 31. SPARQL query to create view](image)

The next step involves creating a workflow that will utilize this view. The goal of the model is to group user with similar battery usage pattern together and find similarity. Using additional information from power screen we can find the time when phone was charging based on screen timeout. For instance if timestamp is late in the night it could give us an idea that the user is sleeping and the phone is charging when screen if off. If the screen is on it mean although the phone is on charge the user is using it. The different charging status throughout the day can give an idea of the users battery level usage.

Due to limited data the results of the model were skewed as it included redundant data. We run the clustering algorithm on the data and created three different cluster based on the attributes provided. We target the battery level, the temperature and the screen timeout status which will help decrypt the pattern. Oracles data miner provides comparison charts that gives us an idea of what the group of users battery pattern is. For example in the cluster 3 and 4 we can see that both the users groups had their battery level constant throughout the month at a particular time period and also the temperature peaked mostly during the mid period indicating the users charged their phones sometime in the middle of the day. However for the cluster 5 we can clear see that the users battery level was always near empty throughout and the temperature level relatively low which mean the users charged their phone at short intervals. From this information one can get the idea that users in group 3 and 4 have similar pattern but users in group 5 differ completely.

![Fig. 32. power battery model](image)

![Fig. 33. Cluster 3 and 4 Result](image)

![Fig. 34. Cluster 3 and 5 Result](image)

![Fig. 35. Cluster 4 and 5 Result](image)

The model helps validate the entire infrastructure and its future application. The model can be more accurate with in depth exploration of different instrumentation provided by phone lab and as well considering a larger userbase.
The main advantage of using this framework was easy construction and storage of complex json data in the form of RDF for the phone lab data. Oracle provides fast and scalable storage solution and hence large amount of real world data can be stored.

VI. CONCLUSION AND FUTURE WORK

In this paper, we worked upon the PLOMAR framework to use incorporate the use of Apache Jena for Ontology models and Oracle data mining capabilities to facilitate management, processing, storing and retrieval of mobile crowd-sensing data. We defined Ontology models for capturing PhoneLabs raw measurement data, constructed new models for representing the correlations between data and created data model queries for deriving context information. We validated our infrastructure with an example to learning a users charging pattern and its correlation with location. From the results it can be concluded that the model is feasible of retrieving this information based on the Ontology models unlike the original JSON format without additional processing.

As an extension to this work, we get permission from phone lab to extend the userbase we can explore in depth the different models. Eventually creating an autonomous model that will generate user pattern based on the changing data.

ACKNOWLEDGMENT

I am thankful to my advisor Dr. Carlos Rivero for his valuable guidance and encouragement during the course of this research, my instructor Dr. Leon Reznik and my classmates for providing me feedback and suggestions. I would also like to thank Rochester Institute of Technology for providing me all necessary equipment and help to fulfill this research.

REFERENCES