A Hybrid Intelligent Agent based Expert System for GPS Databases

Maria Viorela Muntean
Sciences and Engineering Department
“1 Decembrie 1918” University of Alba Iulia
Alba Iulia, Romania
mmuntean@uab.ro
Alexandru Donea
Sciences and Engineering Department
“1 Decembrie 1918” University of Alba Iulia
Alba Iulia, Romania

Abstract – Developing performant mobile expert systems consists an emerging problem taking into account the wide use of mobile phones and the need to access from such devices decision-assistance tools. This paper proposes a Hybrid Intelligent Agent based Expert System for Android which uses a positioning system to enhance its performance.

Keywords – Hybrid Intelligent Agent; GPS data; Expert System; Natural Language Processing.

I. INTRODUCTION

Mobile device has a narrow bandwidth in wireless connection which is a critical problem in developing a mobile information retrieval system [1].

To overcome such problem, mobile agents are proposed, because they could conduct multiple interactions with different information database systems [1].

Recent research on mobile agent networks has frequently been motivated by location-aware services. In such scenarios, the advances in wireless technologies, the spread of powerful portable devices and the development of sophisticated positioning systems enhance the mechanisms for pervasive computing [2].

The location-aware service we focused on in this paper refers to physical location, that is the position of a person, an object, or, in general, a target in the real world. While spatial location is related to a position in the Euclidian space, descriptive location comes to be a common concept in social life, where buildings, roads, cities, territories are identified by names [2]. This is the reason we developed a Hybrid Intelligent Agent based Expert System that get the GPS coordinates of different targets and find the corresponding name in assisting the user in real time. We also added a Natural Language Processing component in order better process the user’s language.

II. RELATED WORK

The paper [1] proposes a GPS Agent as a software agent which can freely migrate from one host to another. First, the GPS Agent captures the user location information from GPS receiver. Then, it migrates from the client (end-user hand-held device) to a remote JADE Tourist Information Center to query the server using SPARQL. At last, the GPS Agent returns to client with related tourist information. In [3] the author focused on controlling an agent in a virtual environment using only GPS information collected with off-the-shelf devices. The papers shows that using a navigation mesh with map matching and A* based path finding, safe routes can always be guaranteed for the agent even though some unnaturalness is sometimes introduced in the paths.

The paper [4] uses an agent-based recognition of relations among people using GPS data to show context information to parent who watch over his child. The recognition functions consisted of autonomous intelligent agents. An agent in lower layer of hierarchy finds the child drop off the way from school to home, then it makes alarm in system.

Our proposed project has been modeled as a platform server architecture, where the agent is connected online to platform via android application and application.

The user will be finally advised regarding the sightseeings that he should visit, taking into account his current position and also will be warned about the targets with negative reviews. The proposed agent works also online, keeping in its knowledge-based the most visited sightseeings, and returns best and safe results from the nearest way to the target.

III. ARCHITECTURE OF THE GPS BASED EXPERT SYSTEM

The proposed GPS Expert System is formed of six blocks (Figure 1):

![Figure 1. Architecture of the GPS Intelligent Agent based Expert System](image.png)
- **Android GPS Sensor**: gets the user coordinates and sends them to GPS Mobile Agent;
- **MySQL GPS database**: stores the coordinates, description and review for each target;
- **Jade Platform**: allow user to communicate with agent using android chat application.
- **GPS Mobile Agent**: waits for queries from user, gets the user coordinates, get targets coordinates and description, executes the rules, returns information to AIML and convert it in natural language;
- **Jess Knowledge Base**: return the nearest and most visited sightseeings, taking into account the user’s coordinates;
- **Natural Language Processing component**: transforms the user’s language into a machine language using AIML.

### IV. WORKFLOW AND IMPLEMENTATION

#### A. Decision process workflow

The decision workflow, presented in Figure 2, describes all the steps performed by the GPS Mobile Agent in order to take decisions regarding the sightseeing and their distances related to the user.

#### B. Hybrid Intelligent Agent Implementation

The Hybrid Intelligent Agent was implemented using JADE platform [5], [6], [7], an open source platform for peer-to-peer agent based applications. Thanks to the contribution of the LEAP project, ad hoc versions of JADE exist designed to deploy JADE agents transparently on different Java-oriented environments such as Android devices [5]. We used this version in order to implement our Hybrid Intelligent Agent that has multiple behaviors: gets user coordinates, gets all sightseeings' coordinates, query MySQL GPS database, calls Natural Language Processing component and executes Jess Knowledge Base.

The GPS Mobile Agent has methods for controlling the time, the calendar, the Android GPS sensor, and informs the user about any change in the area, including weather changes. The Agent connects to a server database which contains details about sightseeings' with positive reviews. The sightseeings' with negative reviews are ignored, because it is very important to help the user in choosing targets to visit.

To receive GPS data, the following Geocoder function was implemented:

```java
location = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(), 1);
lat1 = (double) location.get(0).getLatitude();
lon1 = (double) location.get(0).getLongitude();
```

In order to compute distances from the user to proposed locations, we implemented a `distance` class. Samples of this class are given below:

```java
computeLon = lon1 - lon2;
distance = sin(deg2rad(lat1)) * sin(deg2rad(lat2)) + cos(deg2rad(lat1)) * cos(deg2rad(lat2)) * cos(deg2rad(computeLon));
distance = acos(distance);
distance = rad2deg(distance);
distance = distance * 60 * 1.1515;
//compute distance in km
distance = distance * 1.609344;
```

The attributes `lat1` and `lon1` store the GPS data of the current location, and `lat2` and `lon2` will store the GPS data for target location, after finding out which is this last location by asserting the rules from the expert
system and by querying the database that stores sightseeings’ details and reviews.

Also, Geocoder function from android returns the user location (latitude, longitude, city, state, country, postal code).

```
List<Address> addresses =
geocoder.getLocationByAddress(lat1, lon1);
    address =
    addresses.get(0).getAddressLine(0);
    city = addresses.get(0).getCity();
    state =
    addresses.get(0).getAdminArea();
    country =
    addresses.get(0).getCountryName();
    postalCode =
    addresses.get(0).getPostalCode();
```

The GPS Mobile Agent depends on the GPS system which is connected to the Internet. When it is disconnected from the Internet, the GPS Mobile Agent keeps data offline because GPS gyroscope allows this and GPS enables offline access to data such as latitude and longitude and the Agent keeps browsing history stored in memory until the user is disconnected from the application.

C. Implementation of Rule-based Expert System

Rules in the proposed expert system are represented in JESS, the Java Expert System Shell [8], [9], [10]. The accepted JESS communication acts are:

- **REQUEST** – executed if the sender intends to initiate an action;
- **INFORM** – this informs the receiver;
- **QUERY_IF** – this sends a message and the receiver tries to execute an action;
- **PROPOSE** – sender proposes the receiver to execute an action;
- **CONFIRMP** (call for proposal) – this function calls a proposal for receiver;
- **ACCEPT_PROPOSAL** – executed if the receiver accepts the proposal;
- **REJECT_PROPOSAL** – executed if the receiver rejects the proposal.

For representing the proposals, four JESS templates have been used: user template, city template, sightseeings template and GPS distance template. The proposed templates offer real time information about a GPS position:

```
(deftemplate user
 (slot id_user) 
 (slot name) 
 (slot latitude) 
 (slot longitude) )
(deftemplate city
 (slot name) 
 (slot latitude) 
 (slot longitude) 
 (slot country) )
(deftemplate sights
 (slot type) 
 (slot name) 
 (slot latitude) 
 (slot longitude) 
 (slot city) 
 (slot country) 
 (slot review) )
```

```
(deftemplate GPSdistance
 (slot remainedDistance) )
```

Let us illustrate the approach by presenting some samples of asserted rules. For instance, if the user wants to find targets that are situated at maximum 3 km related to his positions, the following rule will be fired:

```
(defrule sights-distance
 (GPSdistance (remainedDistance < 3))
   => (send (assert (ACLMessage
                     (communicative-act PROPOSE) (receiver
                     ?s) (content "The target is situated at less than 3 km!"))))
```

In order to get targets with best reviews, we proposed a simple rule:

```
(defrule review-target
 (sights (review > 5))
   => (send (assert (ACLMessage
                     (communicative-act PROPOSE) (receiver
                     ?s) (content "The target is very popular"))))
```

When a place is not recommended for visiting, the GPS Mobile Agent fires the following rule:

```
(defrule review-distance-negative
 (GPSdistance (remainedDistance>dis))
   (sights (review ?pop
            &: (> 3 ?dis &: (> 3 ?pop))))
   => (send (assert (ACLMessage
                     (communicative-act PROPOSE) (receiver
                     ?s) (content "This place is not recommended because is situated too far according to your position and it has negative reviews")))
```

To find a certain place, the following rule is necessary:

```
(defrule search-area
 (sights (type = "restaurant"
         | type = "cafe"
         | type = "fast-food"
   && review > 3
   && distance < 3))
   => (send (assert (ACLMessage
                     (communicative-act PROPOSE) (receiver
                     ?s) (content "The results includes fast-foods and cafes in the area!"))))
```

In order to search positive review hotels nearby, the Agent fires the rule:

```
(defrule search-hotel
 (sights (type = "hotel"
         && review > 3))
   => (send (assert (ACLMessage
                     (communicative-act PROPOSE) (receiver
                     ?s) (content "The hotel list will be displayed"))))
```

And for listing details about a target, the following rule is used:

```
(defrule list-historical-target
 (sights (type = "historical"))
   => (send (assert (ACLMessage
                     (communicative-act PROPOSE) (receiver
                     ?s) (content "Will be displayed a new age that contains details about this target")))
```
In the previous presented rules the PROPOSE communication act was used in order propose to the receiver to execute some functions.

The negotiation process has been achieved according to the designed workflow. Examples of asserted rules are presented below:

(assert (sights
  (type "historical monument")
  (name "Union Hall")
  (latitude 46.0221)
  (longitude 23.4311)
  (city "Alba-Iulia")
  (country "Romania")
  (review 6 ))
(assert (user
  (id_user 1)
  (name "userX")
  (latitude 46.0067)
  (longitude 23.5833 )))
(assert (GPSdistance-test
  (remainedDistance 2) //d=2 ))

D. Natural Language Processing component

description

The emergence of natural language interfaces has led to first attempts of programming in natural language [11], [12]. We used some Natural Language processing components in order to facilitate the communication between user (the tourist) and GPS Mobile Agent.

We implemented a function that allows storing conversation into the database and creating an autonomous reply message using ACLMessage function using AIML predicates [13].

AIMLsession=Chatsession.multisentenceResponse;
ACLMessage reply = msg.createReply();
reply.setContent(AIMLsession);
reply.setPerformative(5);
database1.getConnection(UserInput,AIMLSession);
myAgent.send(reply);

AIML and JESS are working together because this allows AIML to ask the expert system which proposal is suitable to be sent.

V. CONCLUSIONS

Providing expert system services through mobiles has many advantages, especially for the tourism domain.

The objectives of this project were to assist and fully inform a tourist about the nearest and most popular sights in a certain city, using a performant mobile application. The user safety is also an important objective of the project, because he will be notified through vibration or sounds when he is next to hazardous areas. The end-user application was built on an Android platform with a simple and efficient human-device interface.

The application database was populated with details about historical monuments, restaurants, tourist attractions, hotels, embassies, or user’s preferred locations stored in memory, even with the safest routes to these destinations.

This paper proposes a Hybrid Intelligent Agent based Expert System which uses a positioning system and GPS data to enhance its performance. The intelligent agent executes a Jess knowledge base and queries a MySQL database in order to advice the user in choosing a certain place to visit. The agent also knows the user coordinates, which are previously sent by the Android GPS Sensor, and also the possible targets coordinates. A distance function was proposed in order to real time compute the distances between the user current position and the targets positions.

A Natural Language processing component was used in order to facilitate the communication between user (the tourist) and GPS Mobile Agent. This makes a series of replaces and finally transforms user assertions into predicates that are better processed by the rule engine.

As a further work, we intend to develop the Natural Language processing component by adding constraints and by unifying predicates.

REFERENCES