Abstract

In this project, a dialogue system was developed using an external toolkit. The system provides navigational instructions in natural language to reach from point A to point B, to users via a web client in uncertain virtual environments. It is a foundational system with plenty of scope for future work. The system was tested in a game environment on subjective qualities, and was found to be a satisfying start.

1. Introduction

Giving Route Instructions in Uncertain Virtual Environments (GRUVE) is a project aimed at developing a dialogue system that renders navigational instructions in natural English language, to a user in a virtual city environment.

This particular system was developed as a participant for the GRUVE challenge. All development and progress was made on top of the toolkit provided by the organizers of the challenge. The toolkit consists of an end-to-end base system, maps, and developer tools to help in the development of the natural language generation module of the system.

In section 2, the toolkit environment and the base system are explained. The developed system was forked from this base system and hence understanding how this system works is the first part of the project. In section 3, study of the base system was performed to find issues and concerns. These concerns were addressed while building the enhanced system. In Section 4, the enhanced system is discussed. In Section 5, the evaluation of the enhanced system is discussed. Section 6 discussed the scope for future work on the enhanced system.

2. Toolkit Environment and Base System

2.1 Introduction

The base system is embedded to a game environment built using Google Street View. The user starts on a street located in Edinburg, United Kingdom. She will help a pirate find and open a treasure chest. The chest, and other required game objects are scattered on streets in the surrounding area, and the user must navigate to each of the objects and finally the treasure chest to win the game. Through the game play, the user has a 'user buddy' (henceforth referred to as 'buddy') available to give navigational instructions that may be useful to complete the game.

The game is played on a web client, like a browser. The user can command the buddy for help via buttons, and drop downs on the game page. The system is hosted on a web server running Java Server Pages and Java Servlets via AJAX requests to generate instructions upon user request, and sometimes periodically.

2.2 System Design

The dialogue system does not support two-way dialogue: the user interacts with the buddy by clicking on buttons, while the buddy interacts with the user through sentences. The system is mixed-initiative, as there are a few ways the buddy can be invoked: a) user presses a button, b) user changes position or orientation, c) periodically once a route request is made. Every time the buddy is supposed to be invoked an AJAX call is made to a servlet. This servlet instantiates the CityModel, the Route Object, calculates the route if a destination is requested and transfers the control to the Dialogue Manager.

The dialogue manager works by using flag bits, to computes the required dialogue act and pass this state to the Natural Language Generation modules. The Natural Language Generation modules return the instruction that is sent back to the web-client for the user to see.

2.3 Dialogue Management

The entire logic of Dialogue Manager is implemented using flags and states. These flags are set and cleared each time the dialogue manager is called. The dialogue manager finally decides on a dialogue act using these flag bits and passes this to the Natural Language Generation modules. The following are the dialogue acts that the dialogue manager may request: greetUser, startWalking, introduceSelf, acknowledge, acknowledgeRouteRequest, acceptThanks, destinationReached, noRouteFound.

2.4 Natural Language Generation

The Natural Language Generation modules are divided into two parts: Lv. 1 and Lv. 2. These modules return one instruction back to the dialogue manager that is carried to the user. Lv.1 of the Generation module handles all dialogue acts but ‘presentRoute’. This is the most important and frequently called dialogue act when the user is being navigated and hence a separate module is called to handle this particular dialogue act. The Natural Language Generation module uses if-else conditional statements to choose an instruction.

2.5 Data Implementation

Most of the data is stored as files that are used by the system. The map of the city is directly rendered using the Google Maps API. Street and Node information is stored in an .osm file from which a CityModel object is built each time. This map files has information about nodes, and their position on the map using Longitudinal and Latitudinal coordinates. Streets are represented as ways, split up by nodes. Each street is stored as multiple ways, each way starting at a node and ending at another. Each street is also given a name that is search for when the routing algorithm runs.

2.6 Routing Algorithm

The routing algorithm is invoked every time the user requests for navigational instructions, or a direction. It computes the route from the current user coordinates to a node that is part of the

destination street. This route is represented as a list of route elements that the generation module uses while generating instructions. Each route element, consists of a node, a way and the name of the street.

2.7 User Interface

The user interacts with the buddy using a set of buttons on the web browser. The following are the functionalities of the each of these buttons in the base system user interface.

Help: The help linked to the dialogue act of introduceSelf, and was replied with a set dialogue each time on how to use the buddy.

Send: The send button is the most used button. The user selects a destination street from the drop down list and clicks on send button. This will open a periodic connection with the servlet and instructions will be served every 10 seconds automatically or when user position has changed.

Stop Directions: This button discards the current route, and stops updating the instructions every time user's position changes.

Which Direction now? : This button is used to make route computations at this current point, and give an instruction. It is seldom used, as route computations are automatically performed every time the user nears a new node. It may come in handy when the user has lost his way and needs a fresh route computation to carry on.

Ok: Button to acknowledge. Not used in any decision making process or responded to.

Repeat Instruction? : Button to repeat instruction. Not useful as the last instruction is always available on the screen.

3. Base System Study

A study of the base system was performed in-house by using the system and logging issues or shortcoming as they appear through the game. The shortcomings and their root cause are mentioned as follows:

1) **Destination detected upon reaching a node, but no information on which side of the node corresponds to the required street.**

Each street has multiple nodes (meeting point of two streets) that it is associated with. When a destination is requested, one of those nodes are chosen. Destination is detected when there are no more nodes in the route list which means and the user is 10 m from the node. To give this information the city model has to be modified to include the side of the node a street is, or calculations can be made to compute this indirectly.

2) **When requested for a destination, the dialogue does not imply a delayed response that is coming.**

The dialogue module had a specific generation rule when the state variable 'acknowledgeRouteRequest' was set. Changes can be made to this rule to send a better instruction.

3) **Sometimes street names are different in the game world and the instruction. Detect street names better.**

Some research on Google Maps revealed that Edinburg, UK where the game is based has a few characteristics that are causing this error. For example, sometimes the same street has two names, street names change at arbitrary points and are not mentioned in the city model. These issues are very difficult to be dealt with.

4) **User buddy sometimes directs the user to a location that is not what is requested, but the names of the requested and directed locations are close.**

For instance, a user request to Clerk Street, routes the user to Glen Street. On further study of the base system and toolkit, it was revealed that the routing algorithm using word distance measure and choose streets that are at a distance of 5 alphabets from the request destination. This seems to be a deliberate feature by the organizers and hence not to be tampered with.

5) **Every 10s the NLG module is polled, and an empty response is sent if there is not request from the user. This case needs investigation.**

When an empty instruction was returned by the generation module, the current instruction was retained, and not cleared. This seemed to be alright for initial versions of the system, but provides scope for small talk with the user as he is navigating the city. Also, this does not affect the navigating that user is involved in as appropriate instructions continue to come when the user reaches the next stage of the navigation.

6) **'null' was sometimes included in the instruction instead of the name of a street. This must be a boundary case, and appropriate steps have to be taken to ensure a viable replacement.**

This was a boundary condition when the user was at the first node in the list of route elements when the route navigating is about to begin. A small tweak in the enhanced system solved this issue.

7) **While the game mentions about the user buddy having a list of options that include people you can talk to, and places you can go to, the list of options does not contain people in the game.**

This issue is out of scope of the project, and this involves adding information in the game world to map characters to positions on the map.

8) **Interestingly the copy of the game that was provided can never be solved. The key is placed on a position in the map, which is inaccessible through Streetview. Thus the user can never get there, and can never over the treasure chest.**

This issue is out of scope of the project, and this involves adding information in the game world to map characters to positions on the map.

4. Enhanced System

Dialogue Generation

The natural language generation module was rewritten. The most primary change was dividing each instruction into three frames: *direction, action* and *entity*.

Direction could take the following values: head straight, head right, turn right, turn around, turn left, and head left. The different between head right, and turn right was used in cases where a street bends, and the user had to head in that direction as opposed to turn into a new street. The value was determined using the angle between the user's current coordinates and coordinates of the next of node in the route.

Action could take the following values: continue on, and walk onto. These values were used to refer continuing on the same street, as opposed walking onto a new street. The value was determined using the angle between the user's current coordinates and coordinates of the next of node in the route.

Entity were usually street names since landmarks were not used in the game world in the base system. Entitles were mentioned based on user's current element in the list of route elements.

The instructions generated in Lv. 1 were changed to be more clear, and continuous. For example, when an acknowledging a route request, the fact that the new instruction will be up in a few moments was added.

5. Evaluation (Enhanced System)

The system was evaluated by three volunteers. A list of subjective questions were asked and rating was given on a scale of 1 to 5, 5 being the best performance. The following five questions were used for evaluation, with the first three being the same as used in the official challenge:

1) **How useful was the buddy in completing the tasks in the game?**

Average Score: 4.34 out of 5 (4.5, 4.5 4.0)

2) **Were the buddy's instructions were easy to understand?**
   Average Score: 4.17 out of 5 (5.0, 4.0, 3.5)
3) **How accurate were the instructions generated by the buddy?**
   Average Score: 3.84 out of 5 (4.5, 3.0, 4.0)
4) Did the buddy give instructions promptly?
   Average Score: 4.57 out of 5 (5.0, 4.7, 4.0)
5) How well did the buddy handle situations when you were lost?
   Average Score: 4.40 out of 5 (4.5, 4.7, 4.0)

6) Do you have other thoughts/suggestions?
   a) Update in instructions required on every move rather than every node.
   b) Ambiguity in street names makes the instruction confusing.
   c) Instruction must update after every move.

6. Future Work

The first version of the system dealt with getting user navigation up and running. The system has scope for numerous enhancements. The following are some of things that future focus will be on:

User Interface

While the use of buttons to interact with the buddy may be the best kind of dialogue in the case of navigation instructions, the option on having the user enter questions is a welcome move and will be investigated in further versions.

Uncertain User Coordinates

Both the base and enhanced system used zero error in user coordinates to determine and route the user to next elements in the route. Uncertainty in user position and orientation brings in new dimensions to the game head right may now be head straight.

Landmarks

When working with uncertain user coordinates, landmarks will come in handy to specify the user conditional instructions to follow.

User modeling

User information is available to the generation module as a list of nodes that the user has visited and the number of times each node was visited, based on the email id the user enters while starting the game. At the outset, it is unclear as to how this information can be used while generating instructions. Further research is required to use this information.

References

[1] Janarthanam, Srini, and Oliver Lemon. "The GRUVE challenge: generating routes under uncertainty in virtual environments." *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, 2011.

Project Code and Setup

The project is housed online at GitHub, here. Find latest setup instructions, project reports, source files, libraries, and dependent install packages in the above link. Setup instructions also available at readme.txt attached along with this report.

1) The Dialogue Manger can be found at:
   source\gruve\WEB-INF\classes\hw\macs\gruve\GruveIM.java

2) The Lv.1 Natural Language Generator can be found at:
   source\gruve\WEB-INF\classes\hw\macs\gruve\NLG.java

3) The Lv. 2 Natural Language Generator can be found at:
   source\gruve\WEB-INF\classes\hw\macs\gruve\HWUNLG.java

Evaluation Volunteers

Name: Ajinkya Waghulde Email id: *awaghuld@usc.edu*

Name: Michael S. Gracias Email id: *gracias@usc.edu*

Name: Nishant Saurav Email id: *saurav@usc.edu*