Natural Language Interface technology in computer games

AN EVALUATION OF NATURAL LANGUAGE PARSERS IN COMPUTER GAMES

DANIEL MALMHAKE, HENRIK KARLSSON
Abstract

A Natural Language Interface is an interface that operates on language input rather than mouse and keyboard. It works by parsing the input into terms, and then parsing these terms into a logic statement that can be processed. The efficiency and viability of Natural Language Interfaces in computer games was compared to the normal input of keyboard and mouse. This was done by having test users play a game with NLI and then running their inputs through a semantic parser. The results indicate that NLI in its current state is less efficient as a way of input than the keyboard/mouse way.

Sammanfattning

1 Introduction

A Natural Language Interface (NLI) is a framework used to control a computer or a similar device through language input, rather than the more common way of manipulating the device through various mechanisms, such as keyboard and mouse. A general NLI works like this: A user creates an input, either through text or speech. This input is then broken down using a link parser. The output of this link parser is in turn put through a Natural Language Parser, connected to a database of semantic rules. Using these rules, the Natural Language Parser translates the syntactic input into an SQL query. The query is then run, and the results of the query are returned to the user. Performance of Natural Language Interfaces may deteriorate due to such things as semantic symmetry. This lowered performance can be compensated with the use of algorithms like shallow parsing (Ekpenyong & Urua, 2011).

In the dawn of the era of computer games, using NLI’s as a basis for user input was one of the more favored strategies in making computer games. Many of the earliest games were text-based adventure games. One of the more famous of these games was “Zork”, which was created in 1977. The game was played by letting the player input natural language commands into a terminal and getting text responses based on what command was entered. Zork was a popular game in the day it was created, but as GUI (Graphical User Interface) technology became more and more advanced, it conquered out NLI’s from the mainstream game scene, and NLI’s were forsaken in favor of GUI’s practicality for larger and more complex games.
2 Objective

In this paper we want to look at whether Natural Language Interfaces (NLI) are suitable for use in computer games. We want to explore the idea of NLI in computer games; do they have a place there, or are they just an archaic or niche approach to getting input from players?

With the growth of the computer game industry an increasing number of game developers have become interested in experimenting with the use of NLI's in games. An example of this is Façade, one of the more popular contemporary games that have been made using NLI technology (Sali, Serdar, Ronald Liu, Noah Wardrip-Fruin, Michael Mateas, and Sri Kurniawan, 2012). It features gameplay where you interact with two characters through natural language text and they react to your actions and behaviour depending on what you do.

We have to be aware of factors that might affect the test results. A text based game differs greatly from the more popular modern games; the most popular games right now are First Person Shooters and strategic games. This possible inexperience might be a hinder for the game’s intended audience. An introductory element, meant to accommodate the user to NLI will have to be introduced, tentatively an easier difficulty of the game.
3 Background

This project will extensively exhaust the subject of Natural Language Interfaces in games. We have both analysed Natural Language Interfaces in current games as well as made our own game which revolves around the usage of a Natural Language Interface.

Besides naming things, specifying numbers or communicating with other players through chat, writing text is something that rarely occurs to players in modern games. Among the top 30 most played games on the steam client, none have Natural Language Interface elements that goes beyond multi-word search queries. We were curious whether this meant that Natural Language Interfaces have become obsolete.

In addition to this, we found it interesting to see how advanced a Natural Language Interface could become.

3.1 Semantic Parser

A semantic parser is a program that takes input data in some form and then interprets the meaning of the data through the use of a grammar. Usually the input gets mapped to a database query that gets returned to the user as an answer to the input. Semantic parsers

3.2 Sempre

Sempre is a toolkit for developing semantic parsers (Berant, J., Chou, A., Frostig, R., & Liang, P. 2013). Developed by a team of computer scientist from Stanford it was created as a tool for constructing general purpose semantic parsers, Sempre functions by creating rules over a set of utterances(input)-answer(output) pairings that depending on the rules used can be combined and create varied parsers capable of recognizing natural language to a high degree. These set of rules are know as a grammar. One way to improve a parser created in Sempre is to train it by providing example utterances together with the answer to them and then let the parser loop through the. The version of Sempre that we use in this paper is sempre EMNLP2013(Berant, J., Chou, A., Frostig, R., & Liang, P. 2013). In EMNLP2013 there is already a grammar available in use that can interpret questions for the freebase database.
Standard CoreNLP
Standard CoreNLP (Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. 2014) is a set of natural language analysis tools that is used to facilitate easier interpretation of raw input text.

Freebase
Freebase is a user-curated database of facts and relations of said facts to each other. The database is free for anyone to use and can be either downloaded or used in an online format.

Webquestions
Webquestions is a dataset of questions with answers that are intended to train the parser to recognize and find the correct answer for queries. The dataset is 3778 questions large and is made in the form of utterances(questions) and targetvalues(answers).

Free917
Free917 is the default dataset that the parser is trained upon in the sempre EMNLP2013 package, It is structured in the same way as Webquestions but is only made up of 641 questions.
4 Method

We created a text based game, where the only way for the player to manipulate the game state is to enter written commands into a prompt box. The goal of the game is to make sure that the button is pressed. Different levels have different difficulties, presented by various obstacles that the player needs to take into consideration while formulating the commands. The intention is that the higher difficulty levels should still be almost as easy as the early levels to solve cognitively, but will require the player to input more commands and thusly force an increased level of NLI usage. The difficulties ranges from a game that almost wins itself to a game which has a solution that is easy to realize but is tricky to formulate. Ideally, every playtester will complete the easiest level on the first or second try and all of them will have to redo or rethink at least one command on the last level.

To maximize user input we decided to implement three different difficulties of the game. This was meant to give different inputs for “different” problems, but also to ramp up the users’ knowledge of the game as well as their experience with NLI. The victory condition of all three difficulties are the same; pressing the button. Since the player cannot manipulate game objects directly, the player has to convince the guard (the role of which is played by our parser) to do it for them.

The only action the player can carry out in the game is speaking. This means that all object manipulation within the game world is carried out through convincing non-player characters to do the players bidding. We want the player to simulate communication with another human being, instead of expressing him/herself in a way one would communicate with a computer program. These interactions is the main focus of the gameplay; we want to make the player think of a broad range of commands to manipulate the non-player characters in different ways to make the player achieve their goal.

The final iteration of the scenarios ended up with three different difficulties; Bob, the easy difficulty, James, the medium difficulty and Earl, the hard difficulty. The easy scenario basically worked as a tutorial for the user/player on how to interact with Natural Language Interfaces; adapting a language like one would in normal conversation is preferred over a more strict “giving a program orders” kind of input. It also served to introduce the idea of doing all gameplay in the form of text inputs. All the player had to do to win was to ask Bob to press the button while using the word “please”.
The medium difficulty served as an introduction of puzzle elements in the game; there
was now a check to see whether James was close enough to the button to press it. To beat the medium difficulty, the player was required to ask James to first move to the button and then tell him to press it. This difficulty could be beat in one or two steps; either by telling James to move to the button and then ordering him to press the button, or by telling him to both move and press in the same command.

Lastly Earl, the hard difficulty, had an obstacle to overcome in addition to the check to see if he was in range. This was a puzzle that required states to be manipulated in the correct order for the player to win. The solution was intended to be easy to figure out cognitively but a bit harder to formulate into words. In order to win on the hard difficulty, the player had to move the guard to a keypad, then enter a very visible code into the keypad, which in turn unlocked the way to the button. Then, after moving Earl to the button, the player could win by simply telling Earl to press the button.

Each level/difficulty was illustrated by a picture intended to give all the information needed for the player to win the level. The pictures where, for Bob/easy;
and finally for Earl/hard, where our intention was that the player should easily link the code together with the keypad, and the keypad together with the glass pane covering the button;

In addition to these images, all levels had an interface consisting of an input field for the player to write in, and an output field for the game to respond to inputs in.
We asked various computer science students to test our game. We gave them the basic instructions “The goal of each level is to press the button, but you can not manipulate anything by yourself. You will have to communicate with the guard on each level in order to make him manipulate the level for you”. Our intention was to give the players as few answers and hints as possible to make the playtesters’ gameplay as autonomous as possible. Our program gathered the inputs for each level and playtester in a text document. Aside from such things as playtesters using synonyms that were not in the lexicon of the game, no one encountered any problems while playing the game. Playing the game to completion on all difficulties took on average about 2 minutes.

For the evaluation of the parser’s ability we created a set of example utterances that we knew would work in the game.

The test set is as follows

“Press the button”
“Walk to the button”
“Press 7541”
“Move to the keypad”
“Press the button please”

The user input was gathered and used to create a dataset of example utterances with corresponding answers that would train the parser to understand input from the game and hopefully enable it to parse the example utterances mentioned before. Since Sempre EMNLP2013 chooses what answer to return depending on a score formula of different weights we can tweak that score formula by training the parser. this is done by giving the system a set of utterances and correct answers and the program then increases the weight for the parts that give score to reach the given answer. We tested the parser by entering the example utterances once before training with the user input. After the training we tested once for every difficulties user input and also once after training with all three difficulties.
5 Results

5.1 User Input

User input was autonomously saved in plain text documents by our game application. Each game difficulty had a different document assigned, and after each playtester had completed testing, the inputs were documented according to the playtesters’ name and what game difficulty the input was from. Each playtester thus generated three separate input documents, one for each difficulty, which generated a good bulk of user inputs. In the table below, we can see the statistics for the user inputs for each difficulty.

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Minimum amount of moves</th>
<th>Expected amount of moves</th>
<th>Average amount of moves</th>
<th>Highest amount of moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy (Bob)</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>Medium (James)</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Hard (Earl)</td>
<td>2</td>
<td>4</td>
<td>10</td>
<td>17</td>
</tr>
</tbody>
</table>

*Table 1: user inputs for the different difficulties*

The Minimum amount of moves (e.g. separate text inputs) is the absolute least number of moves required to win the corresponding difficulty.

The Expected amount of moves is the amount of moves a user is expected to win in by simply playing the game. This is based on things like how informative the image is for the difficulty, and what feedback there is to gain from the game, such as Bob’s response of “ask nicely” if there is no “please” in the input.

The Average amount of moves is simply the combined amount of moves for all testers, divided by the number of testers.

The Highest amount of moves is the highest amount of moves that the difficulty was in by any tester.

An example input;

**NEW USER INPUT:**

insert 7541 on the keypad
keypad bro
keypad bro
keypad
walk to the keypad
write 7541
walk to the button
press that shit
press the button

This is the input from a user playing on the Hard difficulty. What happened here is that the user tried to make the guard input the code into the keypad before moving him to the keypad. The feedback from the program was apparently not as informative as this user desired; we can see the frustration in the repeated inquiries of simply “keypad”. The user made this input despite having learned from previous difficulties that single-word commands are not effective.

After the testing ceased, the gathered inputs were run through the Natural Language Parser.

5.2 Parser Results - Before training

These are results before training the parser on the user input, e.g. the “raw” user input data.
A top value result is a result that the Sempre framework ranked high enough to be judged as valuable and thus returned as a result.

<table>
<thead>
<tr>
<th>Results before training</th>
<th>Press the button</th>
<th>Walk to the button</th>
<th>Press 7541</th>
<th>Move to the keypad</th>
<th>Press the button please</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answers</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>“Telephone keypad 20080726” “A telephone keypad”</td>
<td>0 top value results</td>
</tr>
</tbody>
</table>

*Table 2: The results before training the parser*

“Press the button”
0 top value results

“Walk to the button”
0 top value results
Several candidates but all with a probability less than 0.010

“Press 7541”
0 top value results
“Move to the keypad”
Got the results
“Telephone keypad 20080726”
“A telephone keypad”
“Press the button please”
0 top value results
Highest Probability is fb:en.button (button object) with a probability of 0.461 (too low for a top value)

5.3 Parser Results - After training
These are the results after training the system with the different user input:

Bob

“Press the button”
0 top value results
“Walk to the button”
0 top value results
Several candidates but all with a probability less than 0.010
“Press 7541”
0 top value results
“Move to the keypad”
Got the results
“Telephone keypad 20080726”
“A telephone keypad”
“Press the button please”
0 top value results
Highest Probability is fb:en.button (button object) with a probability of 0.461 (too low for a top value)

James

“Press the button”
0 top value results
“Walk to the button”
0 top value results
“Press 7541”
0 top value results
“Move to the keypad”
Got the results
“Telephone keypad 20080726”
“A telephone keypad”
“Press the button please”
0 top value results
Highest Probability is fb:en.button (button object) with a probability of 0.461 (too low for a top value)

Earl

“Press the button”
0 top value results
“Walk to the button”
0 top value results
“Press 7541”
0 top value results
“Move to the keypad”
Got the results
“Telephone keypad 20080726”
“A telephone keypad”
“Press the button please”
0 top value results
Highest Probability is fb:en.button (button object) with a probability of 0.461 (too low for a top value)

Bob, James and Earl

The result of using all the user input in training

“Press the button”
0 top value results
“Walk to the button”
0 top value results
“Press 7541”
0 top value results
“Move to the keypad”
Got the results
“Telephone keypad 20080726”
“A telephone keypad”
“Press the button please”
0 top value results

Highest Probability is fb:en.button (button object) with a probability of 0.491 (too low for a top value)

<table>
<thead>
<tr>
<th>Results After Training</th>
<th>Press the button</th>
<th>Walk to the button</th>
<th>Press 7541</th>
<th>Move to the keypad</th>
<th>Press the button please</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>“Telephone keypad 20080726” “A telephone keypad”</td>
<td>0 top value results</td>
</tr>
<tr>
<td>James</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>“Telephone keypad 20080726” “A telephone keypad”</td>
<td>0 top value results</td>
</tr>
<tr>
<td>Earl</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>“Telephone keypad 20080726” “A telephone keypad”</td>
<td>0 top value results</td>
</tr>
<tr>
<td>All three</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>0 top value results</td>
<td>“Telephone keypad 20080726” “A telephone keypad”</td>
<td>0 top value results</td>
</tr>
</tbody>
</table>

*Table 3: the results after training the parser*

There were no changes after training the system all answers were the same but with slightly different score on probability for the combined test
6 Discussion

6.1 Hypothesis
Our initial hypothesis was that the technology of Natural Language Interfaces is not useful for games in its current form and capacity. This is because the success rate and precision of the current methods of natural language parsing just isn’t good enough. Compared to other means of user input, it is simply too inaccurate or too cost-inefficient to be a viable option.

6.2 Discussion
For Natural Language Interfaces to have a place in modern games, there is a need for further development. Since the current state of Natural Language Interfaces is to translate one or more sentences into SQL queries, it is practically useless for any task that goes beyond fetching and manipulation of database entries. Perhaps this is why Natural Language Interfaces are almost solely used for menu searching and navigation on modern games. The main part of this problem is that Natural Language Interface elements will never be as efficient as direct input when it comes to manipulation of the game; it takes far more time to formulate a request for gameplay events rather than just doing them via cursor and keyboard-commands. Another problem is that even if we would be able to have a NLP that could interpret most natural language to a high degree of precision, it would require a database far too sizeable and inefficient for use in most modern games. It would most likely have to use an online database to be able to store and handle all the data which isn’t a good solution yet but might be in the future.

We have concluded in the Results section that the Natural Language Parser does not handle our gathered user input very well. What we thought would be an efficient way to handle the Natural Language input in our game turned out to be not nearly as useful as we originally had hoped.

The results show that the training did not impact the way the parser interpreted our example utterances.

The problem with using a Natural Language Parser this way stems from the fact that a game wants extensive feedback as a response; a text-based answer that is perhaps coupled with some parameters changing value.

What the Natural Language Parser is designed to do is basically translating Natural Language input into database fetch-requests and then performing said requests. This presents a great deal of trouble for games that are designed like ours. The input command “press the button please” yields the result of a button object in the database, instead of interpreting it as a command. It tries to interpret it according to the freebase
database, which eventually fails as the parser realizes that it isn’t close enough and it discards the choice. The problem this makes apparent is that the freebase database isn’t suitable for this kind of task, and even when coupled with a training method made from user input it is not enough to construct a parser that can interpret natural language of the form for our game.

7 Conclusion

Since the problem with implementing Natural Language Parsers into games is due to the fundamentals of how most Natural Language Parsers work, we draw the conclusion that this a problem that is present for all Natural Language Parsers. This indicates that the answer to one of our original questions, whether Natural Language Interfaces are obsolete in the modern gaming market, is yes. Natural Language Interfaces can not be implemented in such a way as to sporadically imitate a real conversation. Instead, if they are to be used at all, they should be a core function.
The gameplay of such a game should also be based around the fact that it implements a Natural Language Interface, so as to not become clunky or counter-intuitive. As it seems, Natural Language Interfaces should be considered a gimmick rather than a tool for the player to use in modern games. If a Natural Language Interface is present, it has most likely been the base of development and has had the game built around it, rather than being the best solution to a game-design issue. However, due to the possible factors of errors such as the rough game implementation, no real conclusion can be drawn.
Literary Resources


http://dl.acm.org.focus.lib.kth.se/citation.cfm?doid=2282338.2282353
