Natural Language Interface Technology

IN COMPUTER GAMES

MARTIN MANZANEDA
Natural Language Interface Technology in Computer Games

MARTIN MANZANEDA martmr@kth.se

Stockholm 2015

Royal Institute of Technology
DD143X, Bachelor’s Thesis in Computer Science
Supervisor: Richard Glassey
Abstract

This project delivers the results of research about the implementation of natural language interfaces (NLIs) on computer games. We will study how to combine these concepts, making a small game and evaluating different methods to achieve this.
Sammanfattning

Contents

1 Introduction 4

2 Background 5
   2.1 Overview to Natural Language Processing .............................. 5
   2.2 Natural Language Interfaces ........................................... 6
   2.3 Text-Based Games ..................................................... 8

3 Initial considerations and hypotheses 9
   3.1 Initial considerations ................................................ 9
   3.2 Hypotheses ............................................................ 10

4 Method 11
   4.1 Environment ........................................................... 11
      4.1.1 NLTK .......................................................... 11
      4.1.2 DiffLib module ............................................... 11
   4.2 The game .............................................................. 12
      4.2.1 Scenarios ...................................................... 13
   4.3 Systems ............................................................... 14
      4.3.1 Without NLI implementation ................................ 14
      4.3.2 With NLI without lexical consistency requirement .......... 15
      4.3.3 With NLI with lexical consistency requirement ............ 16
   4.4 Evaluation ............................................................. 17
      4.4.1 Accuracy and effort .......................................... 17
      4.4.2 Usability ....................................................... 17

5 Results and analysis 18
   5.1 Accuracy measurements ............................................... 19
   5.2 Effort measurements .................................................. 21
   5.3 Usability measurements .............................................. 22

6 Discussion and conclusion 24
   6.1 NLI technology in modern games .................................. 24
   6.2 Conclusion ............................................................ 25

7 Appendix 26
1 Introduction

The main objective of this research is to discover how to improve the performance and efficiency of text-based games using natural language interface technologies. Normally in text-based games, we were forced to type exactly the command that the game can understand without mistakes but over the years a research field in natural language interfaces allows computers to understand our language to some extent.

Is it possible to improve usability of text-based games including natural language interface technologies on them?

This research project did a brief explanation about actors that are involved in this study, thinking in possible problems that could happen, getting hypotheses and finally with the development a simple text-based game with natural language technologies where were evaluated the efficiency of different algorithms.

With all mentioned above we can determine to what extent this is realistic in this kind of game, and we whether it is possible to implement on actual games with complex graphics.
2 Background

In this section, the main concepts that will participate in this study will be explained. A brief explanation of Natural Language Processing (field encompassing the NLIs), Natural Language Interfaces and text-based games with the technologies involved in its development.

2.1 Overview to Natural Language Processing

Natural Language Processing (NLP) is an area of research that studies how computers and humans can interact between themselves using natural language text or speech with two main objectives:

- Understand the ability of humans to learn a language, understand and communicate with others.
- Give the ability of computers to understand natural language and act respectfully.

This area must have the presence of a large number of disciplines both technical (computer and information science, mathematics, electrical and electronic engineering, artificial intelligence and robotics) and human (linguistics, psychology). [6]. Machine translation, natural language text processing and summarization, speech recognition, artificial intelligence are easy examples of applications in real life on this field. To achieve this, the computer program needs to address three major problems:

1. The thought process.
2. The representation and meaning of the linguistic input.
3. World knowledge.

The true complexity of developing reliable NLP systems that work at a high level of accuracy may largely be, the complexities and particularity of human behaviour and communication patterns.
2.2 Natural Language Interfaces

A natural language interface is one that allows the user to make query statements or type commands in natural language and get the expected answer in some system, normally on retrieval systems. The main objective of these interfaces is the capability to translate the user statement into understandable actions for the system. At present, much of the efforts in natural language interfaces design have focused on handling rather simple natural language queries due two common problems on this field. [1]:

- The ambiguity, which may have multiple interpretations. E.g. List all students in the school with a Swedish language certificate. Computers don’t know whom to check for the Swedish language certificate - students or school.

- Understanding language often means being able to reason as a human being would reason and having available the full knowledge of a human being, and this problem has proven to be at a level of complexity still far beyond the scope of current technology.

Using Database interaction as an example, it is possible to demonstrate how NLI operates.

Natural Language Interfaces on Databases

Natural language interfaces on databases are systems that allows the user to access information stored in a database by typing requests expressed in some natural language.

![Figure 1: Example of transformation of a natural language question to an SQL query.](image)
Some of most important advantages and disadvantages are listed below:

Advantages

- **No artificial language**: Simply, the user is not required to learn an artificial communication language, in this case, SQL.

- **Better for some questions**: There are kinds of questions that can be easily expressed in natural language, but that seem difficult to express using typical systems, e.g. questions involving negation, or quantification.

Disadvantages

- **Linguistic coverage not obvious**: Users find it difficult to understand (and remember) what kinds of questions the NLIDB can or cannot cope with, e.g. “What are the capitals of the countries bordering the Baltic and bordering Sweden?”, which leads the user to assume that the system can handle all kinds of conjunctions, however this question cannot be handled.

- **Linguistic vs. conceptual failures**: Sometimes it is necessary to rephrase the queries, being more precise and less ambiguous because the system can understand wrongly the sense and respond with other results we expected.

- **Users assume intelligence**: Users often assume that the system is intelligent that it has common sense, or that it can deduce facts and have reasoning abilities. In the other interfaces, the capabilities of the system are more obvious to the user.
2.3 Text-Based Games

A text-based game is a video game that no uses any type of bitmap or vector graphics, all of visual things that compose it are in text and only is possible interact with it typing instructions. This type of games were more popular in the eighties and nineties because these games were typically easier to write and required less processing power, ideally for the computers of that time. However, today some people still play these games. The phrase ”text-based”, made reference to the interface that the user seen instead user input, although these games are generally keyboard-based, the user only seen characters but some text-based games had compositions of symbols such as images. [9]

The two best known variants of text-based games are:

- **Adventure Games**: Are designed for a single player that assumes the role of protagonist in an interactive story driven by exploration and puzzle-solving.

- **Roguelikes**: Are a subgenre of role-playing games, characterized by procedural level generation (sometimes with randomization), turn-based gameplay (that gives the player the time to plan each move), permanent death, and typically based on a high fantasy narrative setting.

There also exists a type of text-based game that allows a real-time online experience between players, the Multi-User Dungeon Games (MUD). At the beginning this kind of games were entirely in text, but then appeared some of these with graphics.

![Figure 3: Zork, a text adventure. [7.]](image-url)
3 Initial considerations and hypotheses

The problem to solve in this thesis is the combination of concepts explained above trying to answer two questions and formulate the hypothesis.

3.1 Initial considerations

Two questions to answer in this investigation:

1. How does a computer game with natural language interface work?

2. Is it applicable to modern computer games?

To answer the first question, the game implements a NLI system, being this kind of games without graphics the easiest and more understandable way to do it because the only way to interact with them is typing commands as actions. The second question is answered after the assessment of procedure and result of the first. The system is tested by a group of people in each different version of the system to be able to measure the accuracy level in each version.

This study considers three different systems in order to understand the difference between concepts seen above:

1. **Without NLI technology**: For the purpose of comparing if implementing natural language technologies in text-based games is better, the user will test the game version without natural language technologies. This system is the original of text-based games.

The main concept in NLIs systems is translate the natural language in understandable commands for the computer. In this case, the game needs an action (the main verb in the sentence) and elements to act (nouns). It will be tested two versions of this NLI system:

2. **NLI without lexical consistency requirement**: The easiest NLI version implementation in the context of text-based games. This version only takes into account the existence of the main verb and the nouns regardless of whether the sentence makes lexical sense or not, e.g. "door open the" instead of "open the door" will work in the same way.

3. **NLI with lexical consistency requirement**: This version is more complex than the second, it needs more accuracy checking the correct words in the sentence, not only finding the verb and the nouns, also looks for a correct lexical meaning.


3.2 Hypotheses

The game where this research studies the implementation of NLI technologies is an adventure text-based game with a beginning and an ending, only with one way to finish it to make it less complex (other adventure games have more than one alternative way to finish), this means that to achieve it the player has to go through different phases in order, e.g. take the keys before being able to open the door. All of these phases have a solution needed to pass to the next, in this system implementation, each solution has the action and the target(s) and to validate it, all that needed is to have these targets available in the player location, e.g. To pass to the next phase, the player has to take (action) the keys (target) but the keys are in the kitchen and the player is in the bathroom. The player can not do that.

This research takes into account each user’s input before find the solution and pass to the next. All of this information is analyzed in four ways:

- **Text similarity:** It compares all the user inputs with the correct solution and then it takes the average in each phase in percentages and then the average of all phases, e.g. Solution: Open the door, Input: Opn th dor, Similarity: 65%

- **Presence of words:** It looks for the words needed for the solution (action and targets), e.g. Solution: [open, door], Input: [open, keys], words matching: 50%

- **Nº of attempts:** It counts the number of attempts before achieve the solution.

- **Time:** It counts the time that player needed to achieve the solution.

And the expected result in each game systems is:

- for the system without NLI implemented, text similarity and presence of words measurements are not so much relevant because the player have to type exactly the sentence needed to solve each phase. In the other two measurement methods, it can compare the improvement with the other systems. It expects be the worst system.

- for the system with NLI implemented without lexical consistency requirement, in all measurements methods, expects the best result because it do not needs too much precision although it sacrifices logic and coherence of the game.

- for the system with NLI implemented with lexical consistency requirement, expects an intermediate result between each others systems in all measurements methods without sacrificing logic and coherence of the game.
4 Method

The following sections describe the method chosen to develop a text-based game and the necessary technologies to develop a natural language interface. Developing a basic text-based game with some locations to go, and some object to interact. Also how to give the capability to translate natural language sentences to actions understandable for the game.

4.1 Environment

The programming language of choice is Python which has a lot of tools to work with natural language like NLTK (Natural Language Toolkit) for tokenization and tagging, and difflib module for check similarity between words.

4.1.1 NLTK

NLTK (Natural Language Tool Kit) is a Python suit of libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, achieving that programs in Python will be capables to understand natural language and work with it. [3]. Specifically, the tools of tokenization and tagging will be very useful in this project because is necessary translate the user sentence in actions understandable by the game.

```python
>>> import nltk
>>> sentence = "I open the garage door with the keys."
>>> tokens = nltk.word_tokenize(sentence)
>>> tagged = nltk.pos_tag(tokens)
```

Figure 4: Example of tokenization and tagging of a user sentence.

NLTK provide us a lexical analysis of the words in the sentence where we can identify, for example, the verb and translate it on an internal action in the game with the following nouns and prepositions with which to link such action.

4.1.2 Difflib module

This module provides a flexible class named **SequenceMatcher** for comparing pairs of sequences of any type. The idea is to find the longest contiguous matching subsequence applied recursively to the pieces of the sequences to the left and to the right of the matching subsequence. This does tend to yield matches that “look right” to people.
4.2 The game

This game is an adventure game with different locations to go to and objects to interact. Normally, in this kind of games, there are characters to speak but to reduce complexity it only have locations and objects. The game system consists in different classes which refer to different elements:

- **Location**: All the Location instances have a title, a description and a list of exits and items inside them. The method *toString()* adds to description all the exits and items for the knowledge of the user.

- **Exit**: The Exit instances are more simple, they only have a direction name, a location instance to go and a parameter that indicates if the exit is locked or not (boolean attribute). The method *toString()* returns the direction name and the name of the location to go.

- **Item**: The Item class has a title and a description but is a little more complex because it needs to know with which other items can interact, storing the name of the action, the target item and the result of this action. The method *toString()* returns the description.

- **Game**: This class englobes all the classes above. Here is where the game logic is designed, creating our locations, exits and items, combining all and creating our history. It have two attributes that control the actual state of the player, current location (Location object) and inventory (list of items). The most important method on this class is the *Action()* method with two parameters, the verb to translate in an action and a list of targets for this action (could be empty if the action do not requires targets).

In this game system, the focus is in the main method of this application where the user input is translated in actions (verb) and targets (nouns) which will be passed to the method called *Action()* in *Game* class mentioned above. For example, the sentence used in the example in figure 4 (I open the garage door with the keys) we can extract the verb "open" and the targets "garage and keys", and call the action method like this: *Action("open", ["garage", "keys"])*. In this case, the action "open" wants two things: the exit name to open and the item that can open it. The result of this action will be change in the Exit object called Garage the attribute locked to false.
4.2.1 Scenarios

The player is in a house with 5 rooms: a kitchen, a bathroom, a bedroom, a living room and a hall.

![Diagram of house layout]

Figure 5: Graphic representation of the main scenario.

The main goal to win the play is exit from the house. To test the various systems, the player will begin in different rooms and each room will have different objects. Each scenario has different ways to achieve out of the room.

Example of a scenario configuration:

**Scenario A**
- **Start location:** Bedroom
- **Bedroom objects:** bed, picture, chair.
- **Living room objects:** coffee table, clock, cabinet.
- **Kitchen objects:** frying pan, microwave, refrigerator.
- **Bathroom objects:** bath, toilet, toothpaste
- **Hall objects:** aquarium, clock, picture frame.
- **Exits:** bedroom/open, living room/open, kitchen/open, bathroom/open, street/open.
4.3 Systems

The next figure shows how is the scheme of the full program. This investigation will focus in the system and it will show how is inside in each version.

![Program escheme.](image)

Figure 6: Program escheme.

4.3.1 Without NLI implementation

The original text-based games were made with this system. The user types one sentence and the game responds, if the sentence is valid, the program will modify itself (changing the current location, adding a item to the inventory, ...), if not, it will respond with the appropriate answer.

It have a ”switch-case” structure with all the possible cases and a default case which will tell the user is wrong whether it can not find any match between the user input and all the possible cases. When the system finds a match with the user input, inside the case statement, it executes all the pertinent actions and responds appropriately.

In Python not exists the ”switch-case” structure but it has used a chain of ”if-else”.

```python
if user_input == "take the keys":
    game.action("take", ["keys"])  
    response = "You have taken the keys."
elif user_input == "use the keys with the door":
    game.action("take", ["keys", "door"])
    response = "You have unlocked the door with the keys."
elif user_input == "open the door":
    game.action("open", ["door"])  
    response = "You have opened the door."
else
    response = "Try again."
```

Figure 7: Example of text-base game system without NLI technologies.
4.3.2 With NLI without lexical consistency requirement

In the systems that implement natural language interface technologies is not needed a "switch-case" structure because the method Action is called directly with the parameters. To achieve this, the system goes through the following phases in order:

1. The user input is processed with the natural language toolkit (NLTK) mentioned above. It splits the input string in tagged tokens, resulting in a list of tuples, e.g. [("open", "VBP"), ("the", "DT"), ...]

2. It searches the tuples with the values VBP (main verb) and NN (noun). Then, it calls the Action method with these parameters.

3. If the action parameter (verb) exist and the targets (nouns) are those it wanted, the system will respond consequently. If not, an error message will be displayed.

This system does not guarantee a correct logic or sense in the game but is an easy way to see how natural language interface technologies can work in a game. The players do not have too many difficulties to finish each phase because they do not have to be too precise.

![Diagram of sentence transformation](image)

Figure 8: Example of a transformation from a user input to an internal instruction. Without lexic checker.
4.3.3 With NLI with lexical consistency requirement

This system follows the same phases as above with a lexicon checker between the first and the second phase. This lexicon checker provides a high fidelity with the game logic, forcing the player to type sentences with grammatical sense. To achieve it, the lexicon checker will follow the next pattern:

\[ VBP[-\rightarrow DT|IN|PRP\$][-\rightarrow NN[-\rightarrow IN|CC |.| |]* \]

This pattern indicates step by step the lexical structure of the user sentences according the logic of the game. The elements in brackets are optional and the asterisk indicates the possibility to appear more than one:

1. Each sentence has to start with a verb (VBP). This verb has to be in present simple of the 1st and 2nd person.

2. Following the verb, there may be or not, one or more nouns (NN). Each noun can have a determiner (DT, e.g. a, an, the), a preposition (IN, e.g. in, with) or a possessive pronoun (PRP$, e.g. his, their). Only it can be more than one noun if there is a preposition, a coordinating conjunction (CC, e.g. and), a punctuation (",.", e.g. ",") between them.

---

Figure 9: Example of a transformation from a user input to an internal instruction. With lexicon checker.

16
4.4 Evaluation

A group of ten testers aged between 20 and 27 years was chosen, and each of them has a minimum of experience in computers and games. They know the instructions to play and the goal of the game before the test, e.g. they know how they can interact with the interface, how to type sentences. To prevent the tester knowing the solution each system has a different scenario, system 1 with scenario A, system 2 with scenario B and system 3 with scenario C. In this manner each measurement won’t be affected wrongly.

4.4.1 Accuracy and effort

To take the accuracy and effort measures it is necessary to know the solution in each game phase. It has to check the text and the words of the solution then if there is a coincidence with all the solution words, the system will check that all the elements are present in the actual player state (correct location and available objects). In accuracy measurements, all percentages (text similarity and matched words) are recorded in each phase. In effort measurements it is taking into account the number of attempts and the total time in each phase. All the solutions have a Boolean flag indicating whether each one is overcomed or not.

Example of a scenario solution:

1. Sentence: Take the keys; Words: take, keys; State: false.
2. Sentence: Unlock the door with the keys; Words: unlock, door, keys; State: false.
3. Sentence: Exit; Words: exit; State: false.

When the player overcomes each phase, the state of that solution will change from false to true and the whole process will start again for the next phase.

4.4.2 Usability

For this measure it will use the System Usability Scale (SUS) originally created by John Brooke in 1986 [4] because it is a standard for quick and easy testing. Each tester will fill this questionnaire when finish the game in each system, answering 10 questions with one of five responses. These questions are a sample:

• I think that I would like to use this system frequently.
• I found the system unnecessarily complex.
• I thought the system was easy to use.
• I think that I would need the support of a technical person to be able to use this system.

The used questionnaire is in the appendix.
5 Results and analysis

There are ten phases with their respective scenarios and solutions. The solutions have the same difficulty level only changing the objects in each room, the starting player location and the actions needed by phase. The next tables show all the results by testers in each system:

<table>
<thead>
<tr>
<th>Tester</th>
<th>Text Similarity</th>
<th>Word Presence</th>
<th>Attempts</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>71.27%</td>
<td>X</td>
<td>29.21</td>
<td>0:19:32</td>
</tr>
<tr>
<td>2</td>
<td>65.35%</td>
<td>X</td>
<td>20.02</td>
<td>0:19:14</td>
</tr>
<tr>
<td>3</td>
<td>62.04%</td>
<td>X</td>
<td>23.95</td>
<td>0:20:19</td>
</tr>
<tr>
<td>4</td>
<td>80.44%</td>
<td>X</td>
<td>30.07</td>
<td>0:16:16</td>
</tr>
<tr>
<td>5</td>
<td>78.26%</td>
<td>X</td>
<td>24.59</td>
<td>0:19:53</td>
</tr>
<tr>
<td>6</td>
<td>45.20%</td>
<td>X</td>
<td>27.70</td>
<td>0:21:38</td>
</tr>
<tr>
<td>7</td>
<td>73.10%</td>
<td>X</td>
<td>30.49</td>
<td>0:19:38</td>
</tr>
<tr>
<td>8</td>
<td>52.41%</td>
<td>X</td>
<td>28.75</td>
<td>0:21:40</td>
</tr>
<tr>
<td>9</td>
<td>46.36%</td>
<td>X</td>
<td>27.07</td>
<td>0:18:04</td>
</tr>
<tr>
<td>10</td>
<td>48.56%</td>
<td>X</td>
<td>30.45</td>
<td>0:19:45</td>
</tr>
<tr>
<td>Average</td>
<td>62.30%</td>
<td>X</td>
<td>27.23</td>
<td>0:19:36</td>
</tr>
</tbody>
</table>

Table 1: Testing of the system without NLI implementation.

It can observe in table 1 that there are not values in word-presence column, this is because this table represents the measurements in the system without NLI implemented that only wants the exact similarity with the solution, there are not tokenization and no matching words.

<table>
<thead>
<tr>
<th>Tester</th>
<th>Text Similarity</th>
<th>Word Presence</th>
<th>Attempts</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36.55%</td>
<td>76.34%</td>
<td>10.38</td>
<td>0:05:25</td>
</tr>
<tr>
<td>2</td>
<td>72.49%</td>
<td>91.52%</td>
<td>5.61</td>
<td>0:06:38</td>
</tr>
<tr>
<td>3</td>
<td>68.54%</td>
<td>82.88%</td>
<td>7.08</td>
<td>0:05:54</td>
</tr>
<tr>
<td>4</td>
<td>32.56%</td>
<td>79.63%</td>
<td>4.82</td>
<td>0:08:08</td>
</tr>
<tr>
<td>5</td>
<td>40.58%</td>
<td>84.58%</td>
<td>3.50</td>
<td>0:08:09</td>
</tr>
<tr>
<td>6</td>
<td>42.35%</td>
<td>82.74%</td>
<td>5.18</td>
<td>0:07:39</td>
</tr>
<tr>
<td>7</td>
<td>53.66%</td>
<td>78.15%</td>
<td>8.56</td>
<td>0:06:59</td>
</tr>
<tr>
<td>8</td>
<td>49.54%</td>
<td>88.67%</td>
<td>8.43</td>
<td>0:06:22</td>
</tr>
<tr>
<td>9</td>
<td>46.21%</td>
<td>78.50%</td>
<td>10.47</td>
<td>0:07:39</td>
</tr>
<tr>
<td>10</td>
<td>39.71%</td>
<td>86.39%</td>
<td>11.95</td>
<td>0:07:37</td>
</tr>
<tr>
<td>Average</td>
<td>48.22%</td>
<td>82.94%</td>
<td>7.60</td>
<td>0:07:03</td>
</tr>
</tbody>
</table>

Table 2: Testing of the system with NLI without lexical consistency requirement.
Table 3: Testing of the system with NLI with lexical consistency requirement.

In the following subsections it compares all average results in each measurement method between systems.

5.1 Accuracy measurements

![Text Similarity Chart]

Figure 10: Similarity averages of each system.
In figure 10, the average of text similarity is compared in each system. It can see that there is not a significative difference between the second and third system (48% y 51% respectively), this is because these systems are in the same condition with this measurement but nevertheless the first system is a little better because to finish each phase is necessary a similarity of 100% raising slightly the average.

In figure 11, it can be seen that the highest average is in the second system (82%) because it is easier for it to find the solution words. The third system (59%) is maintained slightly above its previous measure, keeping it balanced. This measurement is not taken into account for the first system.
5.2 Effort measurements

Figure 12: Nº of attempts averages of each system.

Figure 13: Time averages of each system.
These two measurements in figure 12 and 13 are directly related, when more attempts more time. There is a big difference between the first system and the others, this is because in the first system the player have to type the exact sentence and being forced to try a lot of times. The other two continue in the same way, the second continues offering good results and the third continues stable, always for the same reasons explained above.

5.3 Usability measurements

To understand the tables about SUS questionnaire scores it shows all the questions in the next list:

- **Q1** - I think that I would like to use this system frequently.
- **Q2** - I found the system unnecessarily complex.
- **Q3** - I thought the system was easy to use.
- **Q4** - I think that I would need the support of a technical person to be able to use this system.
- **Q5** - I found the various functions in this system were well integrated.
- **Q6** - I thought there was too much inconsistency in this system.
- **Q7** - I would imagine that most people would learn to use this system very quickly.
- **Q8** - I found the system very cumbersome to use.
- **Q9** - I felt very confident using the system.
- **Q10** - I needed to learn a lot of things before I could get going with this system.

All questions are scored with values between 1 and 5 and to calculate the total score in each questionnaire there are four things to take into account:

- For odd questions: subtract one from the user response.
- For even-numbered questions: subtract the user responses from 5.
- This scales all values from 0 to 4 (with four being the most positive response).
- Add up the converted responses for each user and multiply that total by 2.5. This converts the range of possible values from 0 to 100 instead of from 0 to 40.
<table>
<thead>
<tr>
<th>Tester</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4: SUS testers evaluation for System 1. Total SUS average: 28.25.

<table>
<thead>
<tr>
<th>Tester</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5: SUS testers evaluation for System 2. Total SUS average: 78.75.

<table>
<thead>
<tr>
<th>Tester</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 6: SUS testers evaluation for System 3. Total SUS average: 74.50.
6 Discussion and conclusion

Knowing that this research covers a simplest version of studied concepts, text-based games instead of 3D games and a minimum review of natural language interface technology. It can conclude there is an improvement with the results viewed in section 5 in terms of effort. All users that tested the systems finally preferred those that have natural language interface technologies implemented this can been seen in questionnaire results, the pure text-based game without NLI has a qualification between Awful and Poor while system 2 and 3 have a qualification of Excellent. [2].

6.1 NLI technology in modern games

It can be seen how NLIs can be implemented in the very simplest version of a text-based game starting with a system that did not take into account the lexicon, only the wanted words. This makes no sense in the real world because the games must follow logic but it is good to see how NLI can work together with games and this system provides an easy way to look this. On the other hand, the other system is closer to reality because it follows a logic with a pattern that allows it to find an order and a lexicon sense. However, everything seen is insignificant when compared with video games today, especially those with 3D graphics, where complex commands are required and thousands of parameters and spatial expressions to refer to actions, objects, instruments and locations. In these cases, more accurate parsing is required. e.g. shoot the toxic waste barrel next to the window. [5].

Figure 14: An experimental platform for natural language control. [5]
6.2 Conclusion

Natural language interface technologies can improve computer games but it is difficult to know what kind of game and whether it would be enjoyable for the end user. Therefore, a study should be done before to know whether the development of natural language technologies is the adequate in that moment. The result may vary in completely different ways according to the genre of game and not all of these could assimilate this interfaces in the same way.
### System Usability Scale


<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly disagree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I think that I would like to use this system frequently</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>2. I found the system unnecessarily complex</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>3. I thought the system was easy to use</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>4. I think that I would need the support of a technical person to be able to use this system</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>5. I found the various functions in this system were well integrated</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>6. I thought there was too much inconsistency in this system</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>7. I would imagine that most people would learn to use this system very quickly</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>8. I found the system very cumbersome to use</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>9. I felt very confident using the system</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>10. I needed to learn a lot of things before I could get going with this system</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
</tbody>
</table>
References


