



DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2016

A comparison of interfaces in choice driven games

Investigating possible future applications of NLI
in choice driven games by comparing a menu-
based interface with an NLI in a text-based game

JALDEEP ACHARYA

LUDVIG FRÖBERG



**KTH Computer Science
and Communication**

Jämförelse av gränssnitt i val-baserade spel

Undersöker eventuella framtida tillämpningar av NLI i val-baserade spel genom att jämföra en menybaserat gränssnitt med en NLI i en textbaserad spel

JALDEEP ACHARYA
LUDVIG FRÖBERG

Examensarbete inom datalogi - DD143X
Handledare: Michael Schliephake
Examinator: Örjan Ekeberg

CSC, KTH 2016-05-11

Abstract

Natural language processing has for a long time been a field of research and has been regarded as a thing of the future. Due to its complexity it stopped being featured in computer games in the early 2000s. It has however had a recent revival as a consequence of advancements made in speech recognition, making the possible applications of natural language processing much larger. One market that hasn't seen much in the way of natural language interfaces recently is that of computer games. This report covers the basics of natural language processing needed to implement two versions of a simple text-based adventure game, one with a menu-based interface and one with a natural language interface. These were then played by a test group from which usability statistics were gathered to determine if it is likely that NLP will find its way back in to choice driven games in the future.

The results showed that even though the menu-based interface has a faster rate of progression, the NLI version of the game was perceived as more enjoyable by users with experience in gaming. The reason being that the NLI allowed for more thinking on the user's part and therefore the game presented a greater challenge, something that is perceived as attractive by users with experience in computer games. Also the measured usability was roughly the same for both interfaces while it was feared that it would be much lower for NLIs. Therefore, the conclusion was that it is highly plausible that NLI will find its way back into the gaming world, since it adds a new dimension to adventure games, which is something that attracts users. However, this is given that NLP development continues in the same fast pace as it is today, making it possible to implement a more accurate NLI.

Referat

Datalingvistik, eller NLP, har länge ansetts vara ett forskningsområde som ligger i framtiden, och på grund av dess komplexitet försvann det ur datorspelen i början av 2000-talet. Det har dock nyligen blåsts liv i det som en följd av stora framsteg inom röstigenkänning och röststyrning vilket har gjort att användningsområdena och applicerbarheten av datalingvistiken ökat dramatiskt. En marknad som dock inte har sett så mycket inom NLP på senaste är datorspelsmarknaden. Denna rapport kommer behandla grunderna i datalingvistik och sedan använda dessa kunskaper för att implementera två versioner av ett text baserat äventyrsspel. En version med ett menybaserat gränssnitt och en med ett NLI (från engelskans natural language interface). Dessa två spelades sedan av en testgrupp varifrån statistik samlades för att evaluera användbarheten hos de olika gränssnitten och avgöra om det är sannolikt att NLP hittar tillbaka till valbaserade spel inom en snar framtid.

Resultaten visade att trots att den menybaserade gränssnitten resulterade i att testgruppen avancerade snabbare genom spelet, så uppfattades NLI versionen av spelet som mer underhållande av dem som regelbundet spelade datorspel. Anledningen var att NLiet presenterade en större utmaning i spelet eftersom inga ledtrådar eller alternativ visades av menyn, något som var attraktivt för den vane. Dessutom var den uppmätta användbarheten ungefär den samma för båda gränssnitt trots misstankar om att den skulle vara mycket sämre för NLI. Därför drogs slutsatsen att det absolut finns en chans att vi kan se datalingvistiken komma tillbaka till datorspelsvärlden eftersom det lägger till en helt ny dimension till spelen vilket attraherar användare som letar efter nya upplevelser. Detta är dock givet att utvecklingen av NLP fortsätter i samma snabba takt som den har idag så att mycket mer komplexa NLI:n kan implementeras.

Contents

1	Introduction	1
1.1	Problem Statement	1
1.2	Motivation	1
1.3	Scope	2
2	Background	3
2.1	NLI in games	3
2.2	Natural Language Processing	4
2.2.1	Statistical NLP	5
2.2.2	Corpora	5
2.2.3	Markov models	5
2.2.4	N-grams	5
2.2.5	Natural language interface	6
2.2.6	Stanford Toolkit	6
2.3	Usability Testing	6
2.3.1	Usability	6
2.3.2	System Usability Scale	7
3	Method	8
3.1	The game	8
3.2	Menu-based interface	9
3.3	Natural language interface	9
3.3.1	POS-tagging	10
3.3.2	Corpora	11
3.3.3	Implementation	11
3.4	Testing	15
3.4.1	User testing	15
3.4.2	Test group	16
3.4.3	Game evaluation by users	16
3.4.4	Test logging	16
4	Results	17
4.1	User Testing	17

4.2 Misinterpreted Words	21
5 Discussion	22
5.1 Observations	22
5.2 Improvements	23
6 Summary	26
Bibliography	27
Appendices	29
A System Usability Scale Questionnaire	30
B Game Introductions	31
C Example log file	32

Chapter 1

Introduction

As technology has progressed and computers have become increasingly powerful, new fields of research in computer science have emerged. These are fields that have sprung from the computers' new found ability to perform more computationally heavy tasks as modern processors can perform billions of operations per second. One of these fields is Natural Language Processing (NLP) which concerns a computer's ability to interpret input in the form of a command (as it would be spoken in natural everyday language) and provide the expected output. A natural language interface (NLI) is an interface which uses NLP to process either spoken or written commands.

1.1 Problem Statement

This report will determine if the recent increased performance and usage of NLP is likely to find its way back into choice-driven games. This will be done by comparing a menu-based interface with an NLI in a text-based game, and through means of user-testing gain statistics which will later be used to draw a conclusion.

These statistics will include users' response times which is the time they take between commands, their progression rates through the challenges presented by recording the amount of time elapsed between each key event in the game and the number of commands it takes for them to finish the game. Additionally, the commands written to the NLI will be recorded to be able to measure the performance of the NLI by executing tests such as comparing the interpreted command to the intended command. These recordings will also allow closer examination of what the users expect from the NLI by analyzing the length and type of commands they write. Finally a survey following the System Usability Scale (SUS) standard will be answered by each testee following completion of each version of the game.

1.2 Motivation

There have been many advancements in recent times regarding speech recognition and the parsing of speech to text. This has caused NLP to once again become a

CHAPTER 1. INTRODUCTION

topic of research and discussion since even after translating the speech to text, an advanced NLI is required to parse the text. This is most notably seen in applications such as Apple's Siri, and Microsoft's Cortana. The computer games industry however has not seen much in the way of NLIs recently, even though it was a driving force in NLP research during the 1980's. So while NLP has once again become popular, it has not yet retaken its place among choice driven games. Why it is so is unclear and it is therefore interesting to investigate if there is any future for NLP in choice driven games.

1.3 Scope

The investigation will be limited to choice driven games due to the fact that it is the type of game where one might expect an NLI to thrive as opposed to games that require quick-paced precise actions.

Even though a graphical user-interface would be preferred, it doesn't affect the way the user interacts with the interfaces and is something that could be added to the game later on.

While voice controlled NLIs may be more lucrative, this project will restrict itself to studying NLP with written commands due to the same reasoning as for the GUI. The assumption is made that a voice based NLI would work similarly after an initial step of parsing the spoken words to text.

Finally, a menu-based interface was chosen since that is the original way of playing a choice driven game and an NLI was chosen since that is the interface that is being investigated in this report.

Chapter 2

Background

This section will give some basic introduction to NLP required to understand the contents of this report. It will also present some previous work that has made use of NLIs. However no study comparing NLI to menu-based interfaces in choice driven games has been done to date, and therefore no such work is presented in this report.

2.1 NLI in games

Text-based adventure games were very popular and commercially successful in the eighties[1], but have since gone out of fashion. This change was caused by the parsers that were rather limiting and forced the user into very restricted form of interaction. An example of such a text-based adventure game is Zork[2].

Zork was one of the very first text-based adventure games developed at MIT in the late seventies using something that could be seen as a natural language interface, or NLI[2]. Although, the NLI was a lot more strict and didn't allow complete natural language, instead the commands were restricted to only a few keywords that needed to directly match with one of the provided options for it to work. This still provided a ground for what games like Façade use today[3].

Façade was one of the first games made with NLI so advanced that any command could be written instead of having a limited set of keywords. It was an interactive story of a friend (the player) visiting a bickering couple for dinner. The player could partake in the conversations and discussions of the dinner by simply typing out anything he or she wished to say. The things the player said would then steer the events of the evening, resulting in anything from being thrown out, to helping the couple resolve their problems, to making them fight even more[3]. The NLI worked as such that every command the user typed would be categorized into one of several categories that then affected the mood of the couple[4]. While the NLI used in Façade was primitive compared to those that exist today, it received a lot of recognition since it was one of the few games from that decade that had succeeded in developing a good NLI that the user could interact with very easily. It opened the eyes of many others to the possibilities and scope of NLI in computer games.

CHAPTER 2. BACKGROUND

One of the interesting technologies that Façade made use of was ABL which stands for A Behavior Language and is a reactive-planning language[4].

ABL allows the programmer to create as life-like interactive agents as possible for computer games. Interactive agents are non-player characters in the game with whom the user interacts. These agents must be able to perform several intelligent tasks in parallel which is what ABL provides. Compared with the standard imperative programming languages such as Java or C++ that also can be used to control interactive agents, ABL makes it significantly easier to specify how the agents should act and react. Although, ABL was not used in this project, but could be of interest for someone trying a similar experiment.[5]

Eugene Goostman is one of the most advanced NLI's present till date and provides a third example of how NLI's can be used in computer games. However Eugene Goostman is not a game in the common sense, since there is no goal and there is no way to win or lose. The application is simply an Artificial intelligence (AI) with whom you can converse about anything. It attempts to understand the questions and their context in the conversation to be able to hold a believable conversation with the user. It portrays a 13-year old Ukrainian boy and was the first chatbot to pass the Turing test[6], a test made by the famous mathematician Alan Turing. The Turing test evaluates how human like the AI is by making experts chat with it, without telling them whether they are conversing with a machine or a human. After having spoken for a while the experts try to deduce whether they were chatting to a real person or a chatbot. If enough of the people partaking in the test think they were speaking to a real person instead of a machine, the AI is said to have passed the Turing test[7].

2.2 Natural Language Processing

Natural Language Processing (NLP) is the translation of natural language to input that is understood by a computer. Developing NLP applications is very challenging since computers require the user to interact with them using a language that is very precise, unambiguous and highly structured, something that can be hard to achieve given the characteristics of natural language, which is highly ambiguous and can depend on a number of complex variables.

Current approaches of NLP are based on Machine Learning, which is a part of Artificial Intelligence that deals with pattern recognition in data and using that knowledge in a number of ways to improve a program's ability to interpret input[8]. This will also be the approach used in this study to parse user input, and interpret it using techniques of machine learning. The main techniques of interest for this project will be statistical NLP which will make use of corpora, Markov models, and n-grams.

2.2.1 Statistical NLP

Statistical NLP is a statistical model which is used to reduce the high ambiguity of natural language using realistic grammar, especially when the sentences get longer and more complex. This is often achieved with the use of corpora and a Markov model. A corpus is a database that the NLP program uses to calculate probabilities for the different tasks such as grammar models and a Markov model is a probabilistic model used to calculate probabilities.[9]

2.2.2 Corpora

Corpus is a word derived from latin with the meaning "body" and is used to refer to any text in written or spoken form[10]. However, today it is mainly used to refer to large collection of texts which represent a particular variety or use of the language, mainly used for statistical analysis purposes. There can be many different types of corpora, everything from "General corpora" which consists of general texts that are not related to a specific subject, topic or text type to the so-called "Sublanguage corpora" that are catered to specific requirements and therefore consists of text that sample a particular variety of language. The latter will be used in this project.

2.2.3 Markov models

The simplest form of Markov models are the so called 'Markov chain models' which can be used for statistical description of symbols and state sequences. Something that is interesting about the Markov models is its property of being "memoryless", which means that the probability distribution of the next state depends only on the current state and not the sequence of states that preceded it. This property is normally referred to as the Markov property. The functional principle of this model can be better explained using an example of texts. The states of the model are the words in a lexicon or a corpus, from which the word sequences that are formed are investigated[11]. Using the Markov model, the probabilities can be calculated of a word occurring in a certain context.

2.2.4 N-grams

N-grams are usually used in computational linguistics and probability and is a continuous sequence of n tokens given a sequence of text or speech collected from a text or speech corpus. The tokens can be phonemes, syllables, letter or words depending on the application. N can be any number greater than zero, but the most common ones are 1-gram (or uni-gram), 2-gram (or bi-gram) and 3-gram (or tri-gram). N grams are mostly used for word prediction[13], since they provide a great help in calculating the probability of a certain word occurring after the N-1 previous words. An example of the three most common n-grams are demonstrated in figure 2.1.

CHAPTER 2. BACKGROUND

```
                "I want to go to the park"  
Unigram: {I, want, to, go, to, the, park}  
Bigram: {I want, want to, to go, go to, to the, the park}  
Trigram: {I want to, want to go, to go to, go to the, to the park}
```

Figure 2.1. An example of the three most common n-grams

2.2.5 Natural language interface

A natural language interface is the interface that the user interacts with in an NLP system. The NLI acquires user input which it then parses and provides to the underlying NLP system. If the application requires it, the NLI will also display the NLP systems output.[14] One example of such an NLI is a chatbot which mimics a conversation, where the user is basically chatting with a computer. NLI can be either speech or text-based, but in this study only the text-based variant will be used to better fit the scope of the project. Although, if future application of this study requires speech recognition, all that would be needed is a software that translates speech to text.

2.2.6 Stanford Toolkit

The Stanford Toolkit is a NLP software developed by the Stanford natural processing group available to everyone. It provides important tools to do anything from POS-tagging to deep learning NLP which can be used to solve different kinds of NLP problems. POS-tagging stands for part-of-speech tagging, which basically takes natural language as input and outputs the different part-of-speech tags for each of the words present in the input[12]. A part-of-speech tag is a grammar token, such as noun, verb, adjective etc. Deep learning NLP is an NLP model that isn't task specific, it won't be discussed in any greater depth here however, since it is not used in this project.[15]

2.3 Usability Testing

2.3.1 Usability

One of the purposes of this study being to evaluate the usability of NLIs and comparing them to other types of interfaces, it is important that the term usability and any other keywords used to describe it are clearly defined. There are many different definitions of usability[16], the one to be utilized in this context will be one developed by the International Organization for Standardization (ISO) in 1998, the ISO 9241-11 standard. It defines usability as below:

CHAPTER 2. BACKGROUND

Usability: The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.[17]

While this grants a fairly clear definition of the term it is also important to define the words effectiveness, efficiency and satisfaction. These definitions follow below and are also part of the ISO 9241-11 standard[17]. Worth noting is also the line "...in a specified context of use" which underscores the fact that an application must not have an innate usability, but the potential to be used in a certain environment.

Effectiveness: Accuracy and completeness with which users achieve specified goals.[17]

Efficiency: Resources expended in relation to the accuracy and completeness with which users achieve goals.[17]

Satisfaction: Freedom from discomfort, and positive attitudes towards the use of the product.[17]

Moving forward the words tester and testee will be defined as follows:

Testee: A person playing the game from which data is gathered automatically as he/she plays.

Tester: The person(s) distributing the game to the testees and staying available for questions should any arise.

2.3.2 System Usability Scale

The System Usability Scale (SUS) is a quick and cheap yet reliable way of measuring the usability of a system. It is a ten question poll, each question having a multiple choice answer with five options ranging from "strongly agree" to "strongly disagree". This has become an industry standard since John Brooke created it in 1968 and has been referenced in over 1300 publications.[18] The SUS questionnaire can be viewed in its entirety in appendix A.

Chapter 3

Method

This section intends to describe the ideas behind the implementation of the two different versions of the game as well as how testing data was gathered and treated. It will do so in several steps beginning with the ideas behind the menu-based game, followed by the second version of the game using NLI and finally all thoughts surrounding the testing and the evaluations given by the users.

First and foremost however, an extensive study in literature was conducted. This was done to acquire the knowledge required to implement an NLI, and to research how the application should be tested. The results of this study are presented in the background section of this report.

Java was chosen as the language in which the game was implemented. This was both because of a prepossessed familiarity with the language and the cross-platform compatibilities that allowed easier and wider distribution of the game for testing purposes. The choice of the Stanford toolkit was obvious since it is the most well-known NLP toolkit for Java.

3.1 The game

The game takes the form of the player waking up confused and lost in a dark room. While it remains unclear how the player ended up there, it is quickly revealed that one must escape. The concept of the game is based on Fireproof Games' multi-award winning title *The Room*[20] as it is a popular yet functionally simple game and coming up with a new game would be time consuming without contributing anything to the project but an uncertainty of being as enjoyable. The goal is simply to escape the confinement in which the user finds oneself. Upon escaping one room however another is revealed and the game consists of a total of three rooms before a proper exit is found and the game is completed. There may be several ways of escaping each room but whichever option is chosen is not important, the win condition is simply escaping all three rooms.

To make the comparison between the two different versions of the game as valid as possible, they will both consist of the same three rooms with the same problems

and solutions therein. Thus, differing measured levels of usability for a single user cannot be dependent on anything but the type of interface used.

3.2 Menu-based interface

The menu-based version of the game consists of a series of choices from which the user has to select by typing a number that then corresponds to a certain command. This can be seen in figure 3.1. Once a choice is made, consequences of said decision are displayed on screen and a new set of possible actions are displayed on screen from which one must once again choose. Continuing like so, the player progresses by picking the correct path to escaping.

The difficulty in creating a game with menu-based choices is the difficulty of the game itself. It is hard not to hint at what the player should be doing by leading them on in a certain direction. The step of realizing what a player can do is removed, what remains is only the decision of what the player should do. To avoid the game becoming too easy and a player being able to just randomly select choices to escape, there are many different ways both of succeeding in escaping from a room, but also of failing which results in the player having to start over from the beginning of the current room. Creating more available options for the menu-based game of course carries over to the NLI version as well. This will make the natural language processing feel more thoroughly integrated by providing more available commands, thus increasing the chances of the user typed input to be a valid command.

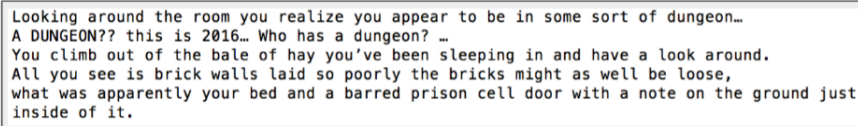
```
Looking around the room you realize you appear to be in some sort of dungeon...
A DUNGEON?? this is 2016... Who has a dungeon? ...
You climb out of the bale of hay you've been sleeping in and have a look around.
All you see is brick walls laid so poorly the bricks might as well be loose,
what was apparently your bed and a barred prison cell door with a note on the ground just
inside of it.
1. Who am I?
2. Read note.
3. Examine walls.
4. Go back to sleep.
5. Walk to the door.
6. Call for help.
```

Figure 3.1. An example of the menu-based interface

3.3 Natural language interface

The NLI is implemented on the same text-based game as the menu-based version, the difference being that the user is no longer presented with available commands but is instead required to type the actions in natural language, although limited to English. This means that when playing the NLI version of the game, a user will need to figure out how to progress for themselves with no hints of what is possible. This is illustrated in figure 3.2.

CHAPTER 3. METHOD



```
Looking around the room you realize you appear to be in some sort of dungeon...
A DUNGEON?? this is 2016... Who has a dungeon? ...
You climb out of the bale of hay you've been sleeping in and have a look around.
All you see is brick walls laid so poorly the bricks might as well be loose,
what was apparently your bed and a barred prison cell door with a note on the ground just
inside of it.
```

Figure 3.2. An example of the natural language interface

3.3.1 POS-tagging

The POS-tagging, or part-of-speech tagging is done using the Stanford NLP toolkit. The tagger is trained by Stanford on a big corpus and has an accuracy of approximately 96%[21]. Although, due to the scale of the English language, the grammar tokens are not only limited to verbs, nouns, adjectives etc, but the tagger also differentiates between the different tenses that these grammar tokens are used in. To limit this to the scope of this project, the tenses were ignored and combined to a common grammar token. The POS-tagger would return a grammar token with one of the following tokens, where the 'other' grammar token is to symbolize all words that don't fit into any of the other categories.

- Noun
- Adjective
- Verb
- Determiner
- Adverb
- Pronoun
- Conjunction
- Other

3.3.2 Corpora

NLI is heavily based on machine learning, so for each possible command that could be executed a corpus had to be created containing every possible thinkable way of trying to express (in natural language) a willingness to perform said action. This means that if twelve commands were available in level one and ten commands in level two, then the corpora would consist of twenty-two corpus files in total. These corpora were initially created manually but later on expanded with users new ways of expressing certain commands, once user testing began. Each file is divided into two parts, the first line in each corpus file consists of the command word and synonyms to it. The rest of the file consists of possible user input in natural language for that specific command. In figure 3.3, an example of a corpus file is illustrated for the command "look around" which would mean that the in-game player should look in their surroundings to see if they can find anything.

```
Look looking around see seeing surrounding nearby
%-%-##-###-%%-%-%-%
Look around
I want to look around
If possible, then I would like to look around
Just look
Start looking
If you dont mind then I would like to look around to see where I am
I want to look where I am
see around
I would want to see around
Just see in your surrounding
Check my surrounding
See if anything is nearby
Can you check for things nearby
```

Figure 3.3. A corpus file

3.3.3 Implementation

The aforementioned corpora were then later used in the first of the two major parts into which the NLI implementation was divided. The first part was about finding the important words from the user input that then were used in the second part to determine which command they corresponded to. Important words are words that are very likely to correspond to a certain command in the game.

To find the important words from any user input, a statistical approach was used. Probabilities were calculated to understand where important words normally occur in a sentence. This was done by calculating the probability of a command word occurring after its preceding word in a sentence and the probability of a command word occurring before its succeeding word in a sentence. Command words are key words that only match to a single command in the game.

CHAPTER 3. METHOD

The probabilities were calculated over all the sentences in the corpora. However, to be able to use these probabilities on sentences never seen before, the probabilities were calculated on the grammar tokens of the command words rather than the command words themselves. To calculate these probabilities, two matrices were created. One that kept track of the probability of a command word's grammar token to occur after its preceding words grammar token, and another that kept track of the probability of a command word's grammar token to occur before its succeeding word's grammar token. This property is similar to the "memoryless" state of the Markov models. An example of a probability matrix can be seen in figure 3.4 where SOL stands for "start of line" and EOL stands for "end of line", which are added to be able to calculate the probability when the command word is at the start or end of a sentence.

	SOL	Noun	Adjective	Verb	Adverb	Determiner	Pronoun	Conjunction	Other	EOL
SOL	x0	y0	z0	a0	b0	c0	d0	e0	f0	g0
Noun	x1	y1	z1	a1	b1	c1	d1	e1	f1	g1
Adjective	x2	y2	z2	a2	b2	c2	d2	e2	f2	g2
Verb	x3	y3	z3	a3	b3	c3	d3	e3	f3	g3
Adverb	x4	y4	z4	a4	b4	c4	d4	e4	f4	g4
Determiner	x5	y5	z5	a5	b5	c5	d5	e5	f5	g5
Pronoun	x6	y6	z6	a6	b6	c6	d6	e6	f6	g6
Conjunction	x7	y7	z7	a7	b7	c7	d7	e7	f7	g7
Other	x8	y8	z8	a8	b8	c8	d8	e8	f8	g8
EOL	x9	y9	z9	a9	b9	c9	d9	e9	f9	g9

Figure 3.4. A probability matrix where the columns are grammar tokens of a command word. The variables would be the probability value in percentage

To calculate the probabilities, the Stanford POS-tagger and bi-grams were used. All the files in the corpora were read line by line and the command word was located in each line. The sentence was then POS-tagged using the Stanford toolkit and divided into bi-grams. The matrices were then updated for the two bi-grams, one that consisted of command word's grammar token and the grammar token of it's preceding word, and the other that consisted of command word's grammar token and the grammar token of its succeeding word. The update consisted of increasing the counter in both the matrices by one, where the row was equal to the preceding/succeeding(depending on which matrix is being updated) word's grammar token and the column was equal to the command word's grammar token. An illustration of this is provided in figure 3.5. After reading through the whole corpora, the values in the matrices were converted to a percentage value.

CHAPTER 3. METHOD

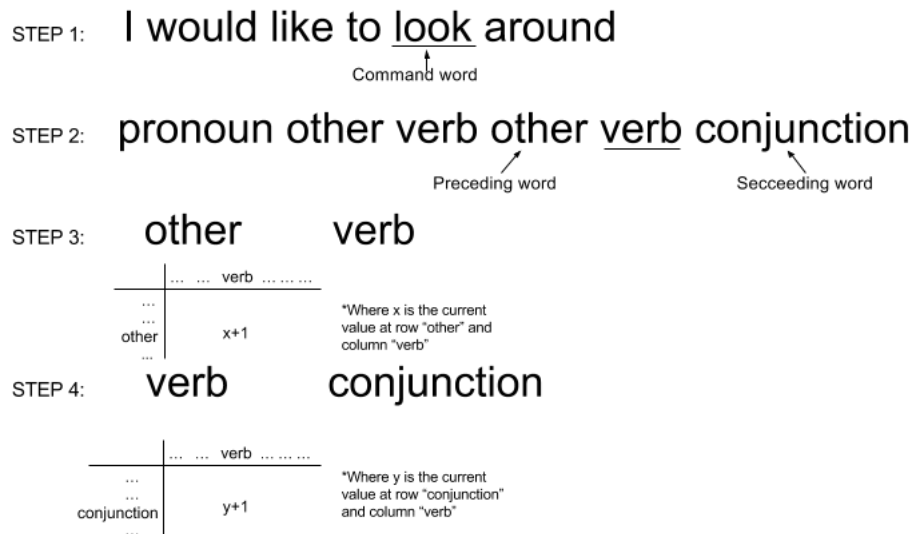


Figure 3.5. Example of the first part of NLI implementation

Using the probability matrices important words were extracted from the user input. This was done by going through the user input and calculating the probability of each word being a potentially important word. The probability was calculated by adding together the value from both the matrices, where the row is the preceding/succeeding (depending on the matrix) word's grammar token and the column is the current word's grammar token. Up to three words were selected, depending on the length of the user input.

The second part was to find a suitable command given the important words from the user input. This was determined by creating yet another matrix, where the rows were the different commands available at the current stage of the game and the columns were the important words. The matrix was then filled by counting the number of times the important words occurred in the commands' corpus files. The values in the matrix were then converted to a percentage value, and all the values from each row were summed together. The command corresponding to the row with the highest value, given that it exceeded a set minimum, was chosen. This is shown in figure 3.6, where command 2 is chosen.

<i>I want to look around</i>			
	<i>look</i>	<i>around</i>	Total
Command 1	1%	6%	7%
Command 2	15%	10%	25% ←
Command 3	2%	7%	9%

Figure 3.6. Example of the second part of NLI implementation

Synonyms

To incorporate the possibility of allowing the user to type synonyms for the command words, a small change was made to the second part of the NLI implementation. When counting the number of times an important word occurred in a certain command word's corpus file, the synonyms to the important word were taken into account, given that they existed in the program. This meant that when counting the occurrences of the important word in a corpus file, the counter was increased even for the synonyms. This can be seen in figure 3.7, where "see" is a synonym to "look" and "surround" is a synonym to "around". As the figure shows, this increases the percentages since synonyms result in a higher count when counting the occurrences in the corpus files. However, command 2 is still chosen.

<i>I want to look around</i>			
	<i>look(see)</i>	<i>around(surround)</i>	Total
Command 1	5%	6%	11%
Command 2	25%	15%	40% ←
Command 3	3%	9%	12%

Figure 3.7. Example of the second part of NLI implementation with synonyms

Avoid Collisions

Problems initially arose when several simultaneously available actions were very similar. For example opening the left door, the right door, or the door behind you when situated in a hallway. All three actions can in many cases be expressed in exactly the same way except for one specific word like left or right. If these then were not chosen as the important words, then the NLI would simply guess which door was to be opened. To avoid this a blacklist was implemented, this meant when selecting important words from the entered commands it was possible to control a list of words that would never be chosen. For example go was blacklisted as almost

CHAPTER 3. METHOD

any command can be phrased with the word go. Some examples of such log entries are displayed below in figure 3.8.

```
256; 2_5; Go open the door
230; 2_3; Go talk to him
516; 3_14; I want to go hide behind the painting
```

Figure 3.8. Examples of how the word "go" can be used for different commands

3.4 Testing

The following sections describe in detail the methods used to gather information to be able to compare NLIs versus menu-based applications. The approach used for testing was highly automated, that is to say it was designed so that the collection of data would be done by the software itself in the background while the user was playing the game. This expedited the testing phase significantly and allowed more data to be gathered quickly. Analysis of the collected data was however done manually afterwards.

3.4.1 User testing

The game was designed to automatically gather all the data that would later be used for analysis. This allowed the testers to spend less time manually recording the events and instead focus on how the user interacted with the interface.

To make the testing as fair and even as possible the test group was divided into two groups. One that would try the menu-based version of the game first, and one that would play the NLI version first. Otherwise the user's enjoyment of the game after already having beaten the game once might have had an impact of the user's perception of the usability of the game. More importantly though, when playing through the game the second time the game is much simpler since the puzzles or challenges are then problems to which the user already knows the answers. Therefore the order in which the users play the different versions of the game must be varied. This will also later be beneficial to the comparison of the NLI with the menu-based alternative as conclusions can be made at a greater extent regarding NLIs both in systems the user is familiar and unfamiliar with.

To further attend the fairness of the testing, testees were prompted to not ask anyone but the testers for help if it was needed. Lastly it was important that each user enter the game with the same information and instructions regarding what it is they are supposed to do. Therefore a short introduction was presented to them in text before each version of the game. These introductions can be seen in appendix B of this report, note that introductions vary slightly also depending on if the game is the first one played, or the second one played. No further instructions were given.

CHAPTER 3. METHOD

The users were never allowed to look at the source code or the logs produced by them playing the game.

3.4.2 Test group

To maximize the amount of data gathered in the tight time-frame allotted, the test group was constructed without being particularly selective or discriminative in choosing candidates. Any and all users that could be found and were willing to test the application were allowed. Effort was made to diversify the test group to gather data from people of varying age, sex, and technical inclination.

3.4.3 Game evaluation by users

A large part of the usability testing was the System Usability Scale (SUS) questionnaire filled out by each user and the resulting SUS grade. This too was automated and built in to the game. The questionnaire contains all the questions and information presented in the paper based version seen in appendix A of this report. However the questions were presented one at a time and the calculations done automatically. This was done both because it decreases the risk of information loss pertained to handing out paper copies of the questionnaire and minimizes the administration and time required of the testers. One word was replaced in the SUS form, it was the word cumbersome that was replaced with awkward as most of our testees spoke English as a second language and found the word cumbersome to be difficult to understand. This replacement is allowed according to data presented by James R Lewis and Jeff Sauro in their book *The Factor Structure of the System Usability Scale*[19].

3.4.4 Test logging

Every action taken by the users testing the game was logged. Each action was defined by a room identifier and an action identifier. Information was also kept on what user input was interpreted as being related to which command and at what time it was performed. The time was measured in the number of seconds elapsed since the start of the game. These time stamps and user inputs were needed to produce the data on how quickly the user reached certain checkpoints in the game, how quickly the user interacted with the system, and to measure the performance of the NLI. Each log was also ended with the SUS score awarded to the system by the player. For an example of a full log file, see appendix C for a randomly selected user test log file.

Chapter 4

Results

The following section presents the relevant data gathered from the user testing performed as described in the method. The data presented here is a product of the log files produced from said testing. This includes how quickly testees were able to perform tasks, how quickly they interacted with the two different interfaces, the performance of the NLI and the results of the SUS form filled out by each testee after completing one version of the game.

4.1 User Testing

The tables and graphs in this section present the data gathered from the logs produced by testees while playing the game. All results have been divided into four different groups. First they were divided into two groups depending on if the testee had played the NLI- or the menu-based version. After that, they were divided with regards to if it was the first or second playthrough by the testee. For all following results a test group of a total of 26 people was used of which half tried the NLI version of the game first, and the other half tried the menu-based version first.

CHAPTER 4. RESULTS

Testee information

	Gamers	Preferred NLI	Preferred menus	Male	Oldest	Youngest	Avg Age	Number of Testees
NLI first	85 %	92 %	8 %	92 %	25	21	23	13
Menu based first	85 %	77 %	15 %	77 %	27	19	22	13
Gamers	100 %	100 %	0 %	91 %	27	21	24	22
Non-Gamers	0 %	0 %	75 %	50 %	23	19	21	4

Table 4.1. Above general statistics and information about the testing group can be seen. The NLI first row represents the statistics regarding those testees that played the NLI version of the game before the menu-based version. The Menu-based first row were those that began with the menu-based interface. The gamers column represents the percentage of testees that considered themselves gamers. The preferred NLI and preferred menus columns present the testees opinions on which interface they preferred after having tried them both. Note that the percentages do not necessarily add up to 100% as some testers considered the two alternatives equal.

Averages

	Actions taken	Average wait time	C1	C2	C3	C4	C5	C6	C7	Finish Time	SUS
NLI as first run	88,88	11,55	101,88	213,63	310,75	462,25	576,50	810,38	894,88	955,25	85,36
Menu as first run	80,11	11,74	107,00	188,00	259,89	497,67	613,33	831,44	890,67	943,89	71,75
NLI as second run	59,56	8,82	108,33	176,33	226,67	315,44	350,89	456,00	494,67	543,56	67,00
Menu as second run	44,00	5,98	35,50	64,13	94,25	132,88	178,50	210,63	244,50	272,88	82,86

Table 4.2. The table displays the average values of each type of playthrough. Actions taken indicates the number of commands issued. Average wait time is the average time between commands. C1-C7 and Finish Time indicate the number of seconds elapsed in average at checkpoints 1-7 and at the time of completing the game. Finally SUS indicates the score awarded to the implementation by the testee after the playthrough on the System Usability Scale. The checkpoints were any key actions that needed to be taken to progress through the game.

CHAPTER 4. RESULTS

Averages NLI

	Longest command	Shortest command	Average command length	% Misunderstood commands	% Not understood commands	% Successful commands
NLI as first run	5,50	1,13	2,60	3,16	4,98	92,07
NLI as second run	4,11	1,22	2,44	1,49	5,72	92,55

Table 4.3. Above is displayed the average statistics specific to the NLI version of the game but compared between players experiencing the game for the first or second time. Longest Command and Shortest Command are the average number of words in the longest and shortest commands respectively issued by the testees. Misunderstood commands is the percentage of commands on average that should have been interpreted as a correct command but were interpreted as an available action other than that which was intended. Not understood commands is the percentage of commands on average that were wrongfully not recognized as any command. Successful commands are the percentage of commands that were recognized successfully as what they were intended to be.

CHAPTER 4. RESULTS

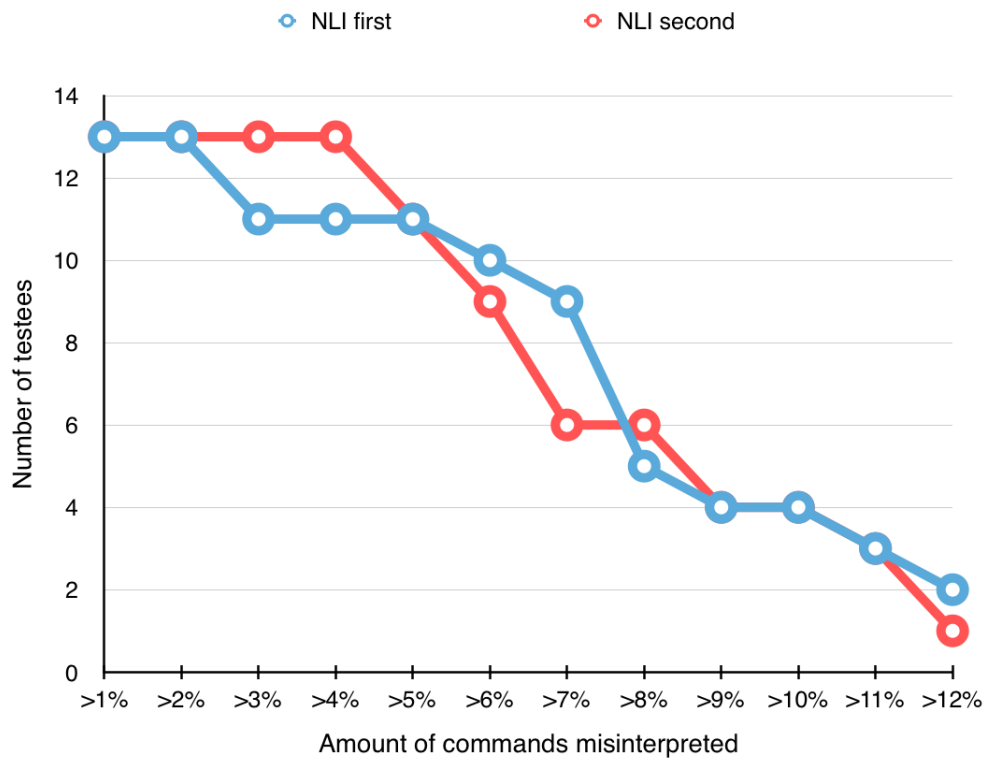


Figure 4.1. Above is depicted the number of testees for the two different groups of NLI testees that experienced more than a certain percentage of misinterpreted commands. Misinterpreted commands includes and is limited to those commands that were wrongfully not recognized as any command at all and those that were wrongfully interpreted as a command other than that which was intended.

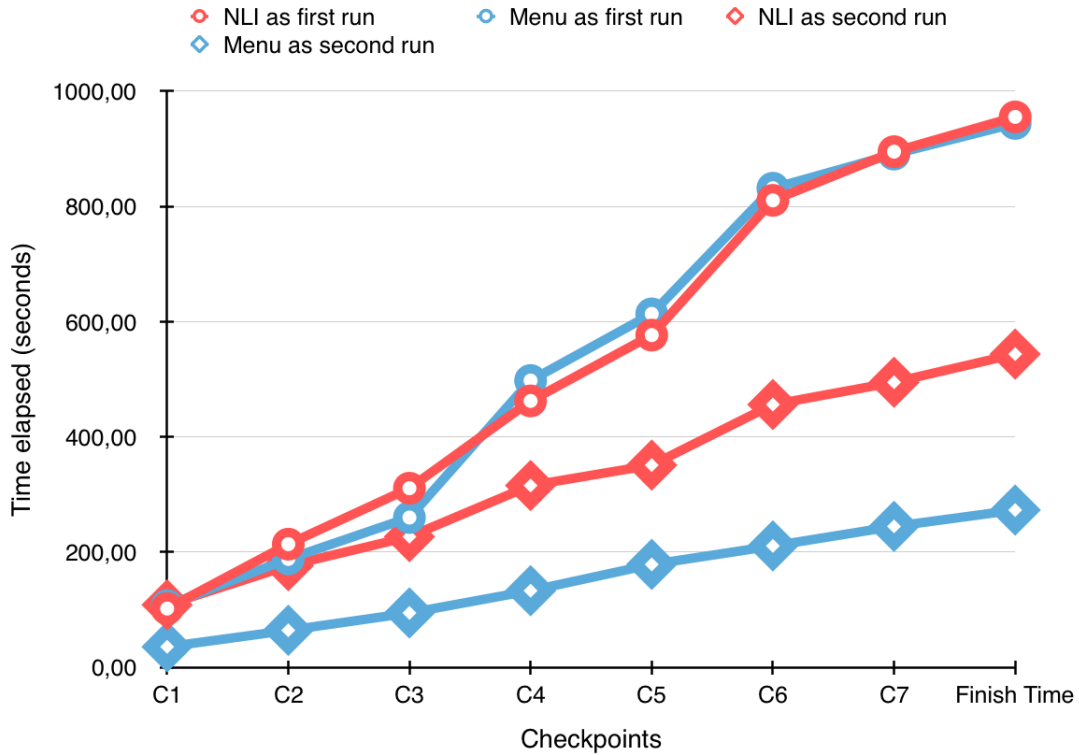


Figure 4.2. The figure above depicts the amount of time elapsed in average for the different types of playthroughs of the game at the different checkpoints (game progressing events). The red lines are the NLI playthroughs while the menu-based runs are blue. Runs that were without previous experience of the game have circles as value indicators while the playthroughs that were done after finishing the game once with the other type of interface have diamonds as value indicators.

4.2 Misinterpreted Words

Issued written commands to the natural language interface version of the game that were not recognized as any valid action were often the result of testees trying to unlock a door before realizing they need to go to the door first to do so. Very seldom was it a result of the user wanting to do something that was actually outside the scope of the game or a proper command not being understood correctly. When a command instead was misinterpreted as a completely different command this was often the result of many simultaneously possible actions being very similar and differing only by a word or two. This was in part solved by implementing a blacklist as described in the method and so this was less of a problem in the later updates of the game than the early ones.

Chapter 5

Discussion

5.1 Observations

The collected results and statistics are at times very indicative of the usability of NLIs compared to menu-based games, but also sometimes not so much. To begin with, looking primarily at figure 4.2, at first glance one might say that the menu-based version is clearly always better than NLI because it is quicker. But what this graph shows is only the time elapsed. More interesting is the difference in the rate at which users interact with the interface, this can be read from table 4.2. There we can see that interaction rates varied very little on the first playthrough and the menu was only 1.5 second faster on the second playthrough. These minor differences might be attributed to the fact that the NLI version requires a bit more problem solving because no alternatives are presented, which may be hinting at how to progress, and the fact that NLI users need to type entire sentences instead of a single digit. The natural language processing also takes about half a second per command issued to the NLI. Therefore the slowed down progression rate cannot necessarily be blamed on usability.

Regarding the usability the SUS scores from table 4.2 should be divided and regarded separately for each group of testers. The group who first played the menu-based version of the game rated both versions of the game lower in usability. If this is because of that group simply not being as positively inclined to the technology overall or if somehow playing the menu-based version first made things feel less usable is unclear. A larger test group would be required to be certain. A clear pattern that can be seen is that both groups awarded a higher average usability score to the interface they first experienced. This indicates that previous experiences play a larger role in a player's perception of the usability of a game rather than the types of interfaces used.

One possible reason why NLIs have not been popular recently in games may be what we see in table 4.2 about the average amount of words in a command. Testees tended to write shorter and shorter sentences as the game progressed and they realized that much effort was not needed to advance. It was seen that most

CHAPTER 5. DISCUSSION

people took the easy and lazy way out once discovering this, as there is no reward for writing longer and more natural sentences. This however defeats the purpose of having an NLI in a game to a certain extent because the language is no longer natural but becomes a series of keywords used to progress faster. At the same time, this still doesn't mean that the game becomes less attractive, since the only thing changed is the fact that user's don't type long sentences. The NLI still requires users to think more to find clues and figure out what to do next, something that challenges the user and therefore makes the experience more entertaining for them. This was noticed when questioning the user about which interface they would prefer when playing a choice driven game, where the majority of the users answered NLI. It also gives the user freedom in the way of interaction, since it allows users to both type short and long sentences, depending on what the user prefers.

Choosing NLI over the classic choice driven approach to game design is a decision that must be made while weighing the benefits of NLIs against their down sides. User testing has shown NLI as more appreciated among seasoned gamers, but those who do not consider themselves gamers find them too complex and difficult. This can be seen most clearly in table 4.1. However, any disapproval may also be due to the specific implementation of the NLI in this game, as seen in table 4.3, where only 92-93% of commands were successfully interpreted as they were intended by the testee. Observe also from figure 4.1 that the amount of commands that could not be interpreted correctly was roughly the same for both people already familiar with what to do in the game and newcomers. This suggests that misinterpreted commands were as likely to occur even when familiar with the system and knowing what to do. In other words any faults in misinterpretation were likely the fault of the NLI implementation rather than a testee not knowing how to proceed in the game. This is something that can be improved when building a game actually intended for the market with an entire studio of developers, money, and years of time behind it. So these misinterpretations need not discourage developers from looking into NLI games either.

5.2 Improvements

The results are entirely based on the user tests which meant that having a diverse group of testers was extremely important. This test group did not consist solely of people who play computer games often, so that an understanding could also be made about how people new to computer games would perceive the NLI. However, the lack of time made it hard to limit ourselves to users of certain age or gender, which meant that the diversity needed couldn't really be achieved. Also, there was an over representation of testees with some kind of a background in computer science or engineering (which meant that many of the testees had had previous experiences or encounters with text-based games), this made the data gathered better representative of how the NLI affected the user experience as opposed to having users who might have had difficulties with using a computer or no experience in playing

CHAPTER 5. DISCUSSION

a text-based game. The inexperience could reflect into some false data since the results wouldn't only be reflective of how the NLI affected their playing experience, but also things such as how well they could figure out what they were doing or how to play the game would also come into play. Since computer-science students are both in age and gender fairly representative of the gaming market[22][23], we believe that it was still a good thing that the majority of the users were from this group. Also a larger test group would have been preferred.

It was considered thoroughly whether it would have been better to allow the testees to play two separate games when trying out the NLI and the menu-based versions of the game. The reason being that since the user plays one of the versions first, it makes it easier to clear the game on the second try using the second version. For example if the user starts by playing the menu-based version of the game, then the user will more or less remember what commands were available when playing the NLI version of the game. No matter how similar in difficulty, playing two separate games would have introduced new ambiguities as well though. It is extremely hard to make two different games as similar as possible when it comes to the playing experience so that the user has the exact same thought process when trying the two different text-based games. So instead of going with two different games, the fact that a second playthrough would be quicker was instead used as an advantage. One common game allowed examination of the performance of an NLI both on a new unknown user, and one that is familiar.

Changes can always be made to the NLI implementation of the game. Taking the results into consideration, one of the common things that was noticed was that even though the user was allowed to type a long sentence, the user mostly preferred to stick to using maximum a couple of words and averaging mostly around 2-3 words. This could be kept in mind while further improving the NLI to better suit the user's way of playing. Some other improvements could include having a larger list of synonyms as well as having a larger corpus. This was done parallel with testing as we discovered testees finding new imaginative ways of expressing their will to perform a certain action. Of course this made later testees' experience slightly different with the NLI than the earlier subjects, however this likely only made the comparison between menu-based interfaces and NLIs more fair over time as the NLI in reality craves much more time to implement, properly and fault free, than the menu-based interface. Something that is important to keep in mind when expanding the corpora is to try to keep the corpus files as unique as possible to be able to easily distinguish between the different command words and minimize ambiguity.

Apart from that, something that is truly difficult to implement is to allow the user to type several commands in a single sentence. An example of such a user input could be "I want to go to the door and read the note". There are several reasons why it would be hard to implement this feature with the knowledge regarding NLP obtained during the course of this project. Since natural language is very ambiguous and the computers require a very precise language, the translation between the two becomes difficult and the only known way today is to use a probabilistic approach.

CHAPTER 5. DISCUSSION

However using the probabilistic approach, the system could figure out that the user wanted to execute two commands and what they are. The hardest part would be figuring out the order in which the commands should be executed. It would also mean that if the system figures out the correct order to execute the commands in, the system still needs to make sure in each step that the command to be executed is valid. If not, then the user needs to be alerted regarding which command that caused the error. However, seeing that the average user input length was 2-3 words, it might be highly unlikely that a user decides to type a long sentence with several commands to be executed in specific order.

Lastly it was realized very late in the testing process that the results could have benefited from having two more test groups. One that played the menu-based version of the game twice, and one that played the NLI twice. These would have made for good references when comparing people playing the game for the second time and would have contained information about testees being familiar with not only the game, but the interface as well and how that affects the way in which the user interacts with the game. This was however not investigated in this project.

Chapter 6

Summary

Improvements can always be made, but observations such as the user always limiting their input to a couple of words even when using NLI can be very useful in future implementation of NLIs in text-based games. The NLI did improve the user experience and made the game more exciting since it required the user to think and figure out what commands they were allowed to use. It therefore made the player more involved which is something that is strived for in the gaming world. However, it also made the game unnecessarily complicated and difficult which can at the same time make the users less interested, especially users with minimum to no previous experience in computer games. Also, since NLP is a very recent topic of research and it is really hard to reach a perfect NLP system, it can lead to some misinterpretations of user input which sometimes can mislead the user playing the game since they might just believe that no such command is possible in the scope of the game, while in reality it was just the faulty NLI.

To conclude, NLI does increase the fun factor of a choice driven game and doesn't significantly decrease the usability as compared to menu-based interface. Given the ongoing fast development of NLP, it is highly plausible that when having a better implemented NLI and text-based adventure game, that the user will prefer to interact with the system in natural language as opposed to menu-based. Therefore it is likely that we will soon again be experiencing NLI in choice driven games.

Bibliography

- [1] Maher, Jimmy. "Let's Tell a Story Together: A History of Interactive Fiction.", 2006.
- [2] Infocom. "Zork", a computer game by Infocom, 1980.
- [3] Procedural Arts. "Façade, a one-act interactive drama.", a computer game by Procedural Arts, 2005. Available at: <http://www.interactivestory.net/technology/> [Accessed 2016 March 28]
- [4] Mateas, Michael, and Andrew Stern. "Structuring Content in the Façade Interactive Drama Architecture.", AIIDE 1 Jun. 2005. Available at: <http://www.interactivestory.net/papers/MateasSternAIIDE05.pdf> [Accessed 2016 April 18]
- [5] Mateas, Michael, and Andrew Stern. "A Behavior Language: Joint action and behavioral idioms.", Life-Like Characters, 2004. Available at: <http://egl.gatech.edu/mateas/publications/MateasSternLifelikeBook04.pdf> [Accessed 2016 April 18]
- [6] Anthony, Sebastian. "Eugene Goostman becomes the first AI to pass the Turing Test, convincing judges that he's a 13-year-old boy", Extremetech, 2014. Available at: <http://www.extremetech.com/extreme/183851-eugene-goostman-becomes-the-first-computer-to-pass-the-turing-test-convincing-judges-that-hes-a-13-year-old-boy> [Accessed 2016 March 29]
- [7] Turing, Alan M. "Computing machinery and intelligence." Mind, 1950. Available at: <http://global.britannica.com/technology/Turing-test> [Accessed 2016 April 1]
- [8] White, Bebo, and Eric Cambria. "Jumping NLP curves: a review of natural language processing research.", IEEE Computational Intelligence Magazine 9.2, 2014. Available at: [http://ieeexplore.ieee.org/focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=6786458](http://ieeexplore.ieee.org/focus/lib.kth.se/stamp/stamp.jsp?tp=&arnumber=6786458) [Accessed 2016 April 14]
- [9] Lauer, Mark. "How much is enough?: Data requirements for statistical NLP.", arXiv preprint [cmp-lg/9509001](http://arxiv.org/pdf/cmp-lg/9509001), 1995. Available at: <http://arxiv.org/pdf/cmp-lg/9509001v1.pdf>. [Accessed 2016 April 18]

BIBLIOGRAPHY

- [10] W3-Corpora project. "Corpus Linguistics: What is a Corpus?", W3-Corpora project, 1998. Available at: http://www.essex.ac.uk/linguistics/external/clmt/w3c/corpus_ling/content/introduction2.html [Accessed 2016 April 14]
- [11] Fink, Gernot A. "Markov models for pattern recognition: from theory to applications.", Springer Science & Business Media, 2014, "Introduction". Available at: <http://link.springer.com.focus.lib.kth.se/book/10.1007/978-3-540-71770-6/page/1> [Accessed 2016 April 14]
- [12] Habernal, Ivan, and Vaclav Matousek. "Text, Speech, and Dialogue: 16th International Conference, TSD 2013, Pilsen, Czech Republic, September 1-5, 2013, Proceedings", Springer, 2013. Available at: <http://goo.gl/x2mCCB> [Accessed 2016 April 15]
- [13] Cornell University. "CS474 Natural Language Processing", Cornell University, 2013. Available at: <http://www.cs.cornell.edu/courses/cs4740/2013sp/lectures/n-gram-models-2-4pp.pdf> [Accessed 2016 April 18]
- [14] Munoz, Rafael, Helmut Horacek, Magdalena Wolska, and Elisabeth MÃ©tais. "Natural Language Processing and Information Systems: 14th International Conference on Applications of Natural Language to Information Systems, NLDB 2009, SaarbrÃ¼cken, Germany 2009, Revised Papers.", Springer, 2009. Available at: <http://goo.gl/iYvnG8> [Accessed 2016 April 10]
- [15] Stanford University. "Stanford NLP Software", The Stanford natural language processing group, 2014. Available at: <http://nlp.stanford.edu/software/> [Accessed 2016 March 05]
- [16] UsabilityNet. "International standards for HCI and usability", UsabilityNet, 2006. Available at: http://www.usabilitynet.org/tools/r_international.htm [Accessed 2016 April 02]
- [17] International Organization for Standardization. "ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability", 1998.
- [18] U.S. Dept. of Health. "System usability scale (SUS)", U.S. Department of Health & Human Services, 2006. Available at: <http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html> [Accessed 2016 March 10]
- [19] Lewis, James R, and Jeff Sauro. "Human centered design", 2009. "The Factor Structure of the System Usability Scale".
- [20] Fireproof Games. "The Room", a computer game by Fireproof Games, 2012. Available at: <http://www.fireproofgames.com/games/the-room> [Accessed 2016 March 20]

BIBLIOGRAPHY

- [21] Roberts, Eric. "Natural Language Processing, Techniques", Stanford University, 2016. Available at: http://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/techniques_speech.html [Accessed 2016 April 13]
- [22] Entertainment Software Association, and Ipsos Insight. "Essential facts about the computer and video game industry: 2015 sales, demographic and usage data.", Entertainment Software Association, 2015. Available at: <http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf> [Accessed 2016 April 15]
- [23] Palm, Therese. "Könsfördelning på tekniska högskolor", 2006. Available at: https://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2006/rapporter06/palm_therese_ [Accessed 2016 April 5]

Appendix A

System Usability Scale Questionnaire

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

Your score is calculated as follows:

- For each odd numbered question, subtract 1 from the score.
- For each even numbered question, subtract the score from 5.
- Add up the new values to a total score and multiply it by 2.5.

You now have your score of 100, a scale of the meaning of each score can be seen below:

92 = Best imaginable
85 = Excellent
72 = Good
52 = Ok/Fair
38 = Poor
25 = Worst Imaginable

Appendix B

Game Introductions

Menu based introduction as first game

Hello,

You are about to play the game "The Escaper" where you wake up to find yourself locked away in an unknown place without memories. You need to escape and to do so you will be issuing commands via a menu based interface. For each action you take new actions will become available numbered 1-xx. Perform an action simply by typing in the number of the action you wish to perform. Good luck!

NLI introduction as second game

Congratulations on completing the menu based version of "The Escaper". You will now play through the game again, however this time you will be using a Natural Language Interface (NLI). What this means is there are no predefined actions presented to you as options. For example if you would wish to look out through a window you would simply type "Go look out the window" or "I would like to take a peek out the window" or any other way you want to phrase it. Good luck!

NLI introduction as first game

Hello,

You are about to play the game "The Escaper" where you wake up to find yourself locked away in an unknown place without memories. You need to escape and to do so you will be issuing commands via a Natural Language Interface (NLI). What this means is there are no predefined actions presented to you as options. For example if you would wish to look out through a window you would simply type "Go look out the window" or "I would like to take a peek through the window" or any other way you want to phrase it. Good luck!

Menu based introduction as second game

Congratulations on completing the NLI version of "The Escaper". You will now play through the game again, however this time you will be using a menu based interface. For each action you take new actions will become available numbered 1-xx. Perform an action simply by typing in the number of the action you wish to perform. Good luck!

Appendix C

Example log file

```
10; 1_0; start by looking around
16; 1_8; read the note
24; 1_4; go to the door
27; 1_8; read the note
37; 1_3; go to sleep
47; 1_8; read the note
54; 1_4; ok then go to the door
58; 1_8; read the note now
85; error ; find needle in the heystack
89; 1_9; go to the haystack
100; 1_2; go to wall
119; 1_7; open box
122; 1_4; open door
126; 1_6; open door
147; 2_1; go to the right
152; 2_4; talk to him
165; 2_0; go to the left cell
170; 2_3; talk to him now
180; 2_5; open the door
196; 2_1; go to john
201; 2_4; talk to john
217; 2_0; kill boris then
221; 2_6; kill boris
237; 2_2; return to my cell
246; 2_9; wake the dog up now
256; 2_8; sneak past the dog
395; 3_0; find light
417; 3_2; go to the table
434; 3_6; smell the drinks
440; 3_7; mix the beers
447; 3_11; go to the painting
480; 3_14; hide behind the paintin
504; 3_15; come out
523; 3_10; go to the hatch
530; error ; go to the table
534; 3_18; climb down
539; 3_2; go to the table
545; 3_7; mix the beers
551; 3_11; go to the painting
```

APPENDIX C. EXAMPLE LOG FILE

```
558; 3_14; hide
562; 3_13; panic
568; 3_13; panic
593; 3_15; come out
601; 3_16; take money from guards
611; 3_10; climb the ladder
617; 3_17; open the hatch
```

```
SUS:::90.0
```

