DEVELOPING A MODULAR EXTENDABLE TOOL FOR SERIOUS GAMES CONTENT CREATION
Combining existing techniques with a focus on narrative generation and player adaptivity.

Master Degree Project in Informatics
One year Level 22.5 ECTS
Spring term 2018

Julian Remi R. Declercq

Supervisor: Mikael Johannesson
Examiner: Per Backlund
Abstract

A large part of any game development process consists of content creation, which costs both time and effort. Procedural generation techniques exist to help narrative generation, but they are scattered and require extensive manual labour to set up. On top of that, Serious Games content created with these techniques tend to be uninteresting and lack variety which can ultimately lead to the Serious Games missing their intended purpose. This paper delivers a prototype for a modular tool that aims to solve these problems by combining existing narrative generation techniques with common sense database knowledge and player adaptivity techniques. The prototype tool implements Ceptre as a core module for the generation of stories and ConceptNet as a commonsense knowledge database. Two studies have been conducted with content created by the tool. One study tested if generation rules created by commonsense can be used to flesh out stories, while the other one evaluated if adapted stories yield better scores.

The results of the first test state that adding rules retrieved through common sense knowledge did not improve story quality, but they can however be used to extend stories without compromising story quality. It also shows that ideally, an extensive natural language processing module should be used to present the stories rather than a basic implementation. The statistically insignificant result of the second test was potentially caused by the compromises taken when conducting the test. Reconduction of this test using real game data, rather than data from the compromised personality test, might be preferable.

**Keywords:** Content Creation, Procedural Narrative Generation, Modular Tool, Player Adaptivity, Adaptive Narratives
Acknowledgments

I would like to express my gratitude towards my supervisor, Mikael Johannesson, for the group, as well as the individual sessions, concerning my thesis. The brainstorms and discussions provided throughout those contained a lot of useful ideas and have guided me in the right direction. I would also like to thank my examiner, Per Backlund, for the invaluable feedback he provided during the mid-semester seminar. His remarks sharpened my focus and prevented me from getting too deep into off-topic issues.
# Table of Contents

1 Introduction 5

2 Background 6

2.1 Serious Games 6
2.2 Player experience and adaptivity 6
2.3 Stories in games 7
2.4 Procedural content generation 9
2.5 Commonsense knowledge 10

3 Problem 12

4 Method 14

4.1 Modularity 14

4.1.1 Ceptre module 15
4.1.2 Common sense module (ConceptNet & WordNet) 15
4.1.3 Language interpreter module 15
4.2 Evaluation of the tool 17
4.3 Ethical considerations 17
4.4 First study: story generation, common sense and language interpretation 18
4.5 Second study: adaptivity through the Ceptre module 18

5 Procedural narrative generation 19

5.1 Narrative adaptivity research 19
5.2 Recurring elements 20

5.2.1 The initial state or context 21
5.2.2 The progression or evolution 22
5.2.3 The debriefing or evaluation 22
5.2.4 Conclusion 23

6 Ceptre, a rule specification language 24

6.1 Non-determinism 24
6.2 Debugging 25
7 Study: Story generation, common sense and language interpretation 27
  7.1 Implementation 27
  7.2 Procedure 29
  7.3 Presentation 29
  7.4 Evaluation method 30
  7.5 Results and discussion 31
8 Study: Adaptivity through the Ceptre module 33
  8.1 Participant feedback 33
  8.2 Procedure 34
    8.2.1 First component 34
    8.2.2 Second component 34
    8.2.3 Story selection 35
  8.3 Evaluation method 36
  8.4 Results and discussion 37
9 Conclusions 39
  9.1 Summary 39
  9.2 Discussion 39
    9.2.1 Positive remarks 40
    9.2.2 Negative remarks 40
    9.2.3 Research objectives 40
  9.3 Future Work 41
10 Appendices 48
  A - First study Ceptre code without the addition of common sense knowledge 48
  B - First study Ceptre code with the addition of common sense knowledge 51
  C - Simple language interpreter module example 54
  D - Stories presented in Study: Story generation, common sense and language interpretation 56
  E - Stories presented in Study: Adaptivity through the Ceptre module 58
1 Introduction

It is easy to see the pain and suffering caused by physical diseases or physical trauma, it is not so easy to see the mental suffering of trauma or severe mental illness. But what if we could find a way to see it? Games are capable of drawing you in for hours on end, playing the role of a character who’s different from you, experiencing their perspective, and actively involving you in a world that functions with a different set of rules.

Antoniades T., founder of Ninja Theory (2017)

It is undeniable that games are a powerful medium of conveying messages. Serious Games in particular focus on this non-entertainment part of games. Narratives in games are important, their use ranges from working as a backbone for the game to being the most prominent element of an experience where the game works as a backbone for the narrative.

Creating narratives takes time however, lots of time, and lots of effort. Stories written by human authors are carefully crafted to have a certain pace, dramatic tension, foreshadowing, narrative arc, etc. (Shaker et al., 2016). Serious Game developers tend to operate in much smaller and more intimate environments than their commercial counterparts. Creating the scenarios, levels and narratives in advance is a major effort and consumes a large part of the development budget (Yannakakis et al., 2009).

Procedural generation can help resolve this problem. Instead of spending multiple days or months of extensive manual labour, complete environments can be created from algorithms that were given merely initial descriptions or even just random generator seeds. However, when done incorrectly, this procedural content can lead to boring, impersonal limited training scenarios that prevent the true potential of the game to express itself (Lopes, 2010). When procedural generation is not done right, a lack of variety in content arises (Charles et al., 2005). A hybrid system that combines best of both human-authored and computer generated worlds is thus a highly desirable approach.

This thesis aims to provide a prototype for a tool that combines existing techniques for narrative generation and content/player adaptivity. By doing this it tackles the current problems concerning extensive manual labour and generic, impersonal and uninteresting Serious Games.

The studies described in this thesis are conducted to find out if common sense databases can be used to extend or create new rules for narrative generation and if storylines adapted to the player yield better subjective scores. The whole tool and its modules are meant to be expanded upon and everything coded in this thesis is completely open-source, available on GitHub (Declercq, 2018).
2 Background

2.1 Serious Games

Serious Games is the collective name for all games and applications that represent a purpose next to pure entertainment. They aim to teach the user by making use of the, possibly hidden for the player, pedagogical value of fun and games. The applications of Serious Games go beyond academic purposes, they are spread around a wide array of fields such as military simulations, healthcare, corporate programs and cultural games (Frutos-Pascual and Zapirain, 2015).

Marsh (2011) illustrates the diversity of Serious Game as well. The author proposes an approach that categorizes Serious Games through a sequence of earlier established gaming characteristics such as challenge, play and fun. The categories range from video games with a purpose at one end to environments with minimal traditional gaming characteristics on the other. The latter’s main purpose is to provide experience and emotion in order to convey meaning. In short, it can be said that digital or traditional Serious Games have one thing in common, which is serving a purpose next to entertaining. Whether that purpose is education, persuasion (leading a learner towards a certain conclusion) or just grabbing attention towards a specific subject, is of little importance when classifying a game as a Serious Game.

2.2 Player experience and adaptivity

Player experience is crucial to Serious Games because they try to enforce the player into learning or directing their attention to a certain topic, regardless of whether the player is conscious of this. Engagement, which is linked positively to desirable learning outcomes (Carini, Kuh and Klein, 2006), is harder to achieve in non-entertainment focused games, which Serious Games falls under, when compared to games that are focused on entertainment.

As the word itself suggests, player experience is highly dependent on the players themselves. It varies across players and even across play sessions. Every player is different, each have their own preference for pace and style gameplay-wise. On top of that, game playing capabilities and skills between players can vary widely (Charles et al., 2005). Developing a (Serious) game that connects to the whole spectrum of players, which ensures an optimal player experience and thus learning process, is a very complicated issue.
This is where player adaptivity comes in handy. It is possible to dynamically tailor a game to individual players by using player modelling techniques and adaptive game technologies (Charles et al., 2005). Charles et al. propose adaptive game technology as a possible solution for the so-called “sameness” and lack of variety in gameplay.

Lopes (2010) points out that the Serious Games research area currently suffers from a lack of systems which have the ability of adapting storylines and narratives to the player and don’t do this at the cost of accessibility. A direct consequence of this lack is Serious Games often ending up with generic, impersonal, limited and stereotyped training scenarios that prevent the true potential of the game to express itself (Lopes, 2010). If every player of a Serious Game ended up playing the game as intended by the designer, the teaching effect would be optimal. Considering this is not the case however, Serious Games could benefit greatly from approaches that include content adaptation.

Recent years have witnessed increased interest in the use of interactive language technology in educational and entertainment applications. Computational storytelling could play a key role in these applications by effectively engaging learners and assisting them in creating a story. It could also allow teachers to generate stories on demand that suit their classes’ needs. [...] Ideally, the plot should adapt dynamically in response to the players’ actions.

McIntyre & Lapata (2009, p. 217)

Above quote, abstracted from McIntyre & Lapata (2009), shows that there is potential for computational storytelling to play a key role in engagement for learners. Easily accessible systems that can generate, as well as adapt storylines to the player, could definitely benefit Serious Games.

Not only does this save a lot of work for the designer, it also ensures that every player has a better experience with the game and it is thus highly likely to have a positive effect on the learning experience.

### 2.3 Stories in games

Storylines and narratives differ in use throughout different categories of games. In some games they are used as a mere backbone for motivation. Their main purpose is to set the general theme and mood of the game, in order to motivate exactly why the players are doing what they are doing (Shaker, Togelius & Nelson, 2016). In this case, where the narrative of a game acts purely as a backstory, the level progression and game mechanics have little to nothing to do with the storyline that happens and develops after the start of the game. The well-known Super Mario Bros game(s) can be taken as an example. The protagonist plumber, Mario, traverses the Mushroom Kingdom in order to rescue Princess Toadstool (more commonly known as Peach later on) from the antagonist of the series, Bowser. This
very limited information is enough to motivate the player to complete multiple levels while avoiding different threats, and it serves no further purpose.

Games such as *Hellblade: Senua’s Sacrifice* have it the complete other way around when compared to games where the story works as a backbone for the game. Here the game works as a backbone for the story instead. In this particular example, the game in its entirety acts as a metaphor for the psychosis of which the main character suffers (Ninja Theory, 2017). Ninja Theory, the developer team behind *Hellblade: Senua’s Sacrifice*, has consulted world-leading neuroscientists and non-profit organizations like Wellcome Trust in order to properly capture the experience of psychosis and its devastating effects on the human mind (Briers, 2017).

The developer went as far as including a 25-minute documentary inside the game that addresses their mental health research on the topic of psychosis in general. In the video, professor Paul Fletcher, who is a psychiatrist and professor of health and neuroscience at the University of Cambridge, displays a great interest in the way Ninja Theory portrays mental illness, something he explores scientifically, in their game. He believes it might offer insights that could not have been retrieved through pure scientific explorations, on top of retrieving a quite unique empathic view of what psychosis might be like (Ninja Theory, Fernyhough and Fletcher 2017).

![Figure 1](image.png)

**Figure 1**  Hellblade Senua’s Sacrifice

Charles Fernyhough, professor of Psychology at Durham University, who has developed a special interest for hallucinations and who is the author of *The Voices Within*, a book with the main topic being a recovering psychosis patient, can be seen vouching for the approach of the developer. Underlying quote clearly hints, possibly unconsciously, that *Hellblade: Senua’s sacrifice* is a Serious Game with as goal raising awareness on psychosis.

I hope that others will follow the lead that they [Ninja Theory, the Senua’s Sacrifice development team] have set and use the power of something like a video game to change people’s perceptions, to increase understanding and ultimately to make some of the stigma around voice hearing and other experiences go away.

Ninja Theory, Fernyhough, C., Fletcher, P. (2017)
This hint towards *Hellblade: Senua’s Sacrifice* being a very well executed Serious Game is confirmed in the full in-game documentary (Ninja Theory, 2017). In the documentary, the founder of the company states that the game was always supposed to hint to the player that it was taking place in the protagonist’s mind, but originally the narrative wasn’t centered on her mental illness. However, the team saw the potential and possibility of this idea and rose to the occasion, which can be derived from the quote below.

It is easy to see the pain and suffering caused by physical diseases or physical trauma, it is not so easy to see the mental suffering of trauma or severe mental illness. But what if we could find a way to see it? Games are capable of drawing you in for hours on end, playing the role of a character who’s different from you, experiencing their perspective, and actively involving you in a world that functions with a different set of rules.

Antoniades T., founder of Ninja Theory (2017)

For completeness’ sake it should be mentioned that another form of narrative design behaviour exists in between both aforementioned types. This type also tackles the shortcomings of the storyline as merely a backbone and can be seen in game franchises such as *Final Fantasy*. Here, the gameplay takes place in the provided narrative arc. In this case the storyline is also of more importance to the game than just an initial setting or moodsetter. These categorisations from Shaker, Togelius & Nelson (2016) could better be interpreted as a spectrum rather than fixed categories.

### 2.4 Procedural content generation

Procedural content generation can be used to generate great amounts of content from minimal input. Whether for narratives or not, procedural content generation is the act of creating data algorithmically as opposed to manually. It offers a great potential for multiple aspects in games. Due to the small or often non-existent need of human input, it can create vast amounts of different output and thus proves very useful as a tool for creating or expanding games. The ample amount of content that can be created using procedural generation is not the only major advantage. Applicable to the field of Serious Games for example, the generation process can be modified to adapt content, in real-time or not, to the player’s proficiency in a specific part of the game. This personalised content can lead to the improvement of the learning process of the player as has been discussed in *Player experience and adaptivity*.

In most computer games the scenarios, levels, storylines etc. are almost completely made in advance. Creating all this game content is a major effort and consumes a large part of the development budget (Yannakakis et al., 2009).
The Siren project proposed by Yannakakis et al. (2009) does however, as most Serious Games, not possess the vast budget to afford the luxury of having professional game designers create an extensive amount of scenarios that include all possible combinations of players and/or player groups. Instead, it is preferable to take a different approach: developing mechanisms for automatically generating and adapting scenarios to given specifications and constraints.

It is because of these reasons that a number of games implement some way of procedural content generation where complete environments are created from algorithms that were given compact initial descriptions or even just random generator seeds (Yannakakis et al., 2009).

A specific in-depth discussion for procedural generation for narratives in particular is provided in the section Procedural narrative generation.

2.5 Commonsense knowledge

Commonsense knowledge is the knowledge that consists of facts about the everyday world that, as the name suggests, are expected to be common sense and thus known by all humans. The idea of commonsense knowledge dates back to 1959, where John McCarthy proposed a hypothetical program called Advice Taker (McCarthy, 1959). Despite the early birth of the idea, the implementation itself didn’t take place until roughly 40 years later, with the Open Mind Common Sense project (Singh et al., 1999).

Every ordinary person has the common sense we want to give our machines.

Singh et al. (2002)

Above quote demonstrates again the ideology that is hidden behind the word commonsense knowledge. Singh et al. (2002) describe the origin of the original Open Mind Common Sense system, abbreviated as OMCS-1, and how it has been online and running on the web since September 2000. Roughly two years after its creation, in 2002, it already contained more than 450,000 pieces of commonsense knowledge such as facts, rules, stories and descriptions. All of these were retrieved using a variety of simple elicitation activities (Singh, 2002). All data was retrieved from over 9000 people, many with no special training in computer science or artificial intelligence. The latter fits in well with the description of the ordinary person from the mindset displayed in above quote.

Commonsense knowledge lends itself as a nice first module, the definition of a module is further explained in the Method section, to implement for the prototype application proposed in this thesis. Liu and Singh (2002) were one of the first to implement an approach using commonsense knowledge with their system MakeBelieve. MakeBelieve takes an initial sentence from the user as input and generates short stories with a length of 5 to 20 sentences as a result. It does so by discovering likely consequences to the actions present in the given
sentence. It compares similar words in the current sentence and does a lookup in the database for possible consequences. A nice example for this logic is demonstrated by McIntyre (2011) where the input sentence “Mary broke her leg” results into the follow-up sentence “Mary went to the hospital”.

The knowledge from ConceptNet (Speer and Havasi, 2012), one of such knowledge bases, was built by automatically extracted data from human authored sentences that were collected as part of the earlier described Open Mind Common Sense project (Singh & Barry, 2003). McIntyre (2011) has proven the possibility to, to a certain extent, finish incomplete stories through integrated commonsense knowledge used to provide additional information about the story entities and actions.

It should be mentioned that commonsense knowledge has one major, and quite obvious, drawback which is the dependence on the information available in the database. The quality of the resulting sentences retrieved through a commonsense knowledge approach can only be as good as the quality and amount of data provided by the database.
3 Problem

Stories written by human authors are carefully crafted to have a certain pace, dramatic tension, foreshadowing, narrative arc, etc. (Shaker et al., 2016). However, for video game narrative generation, story planning from the perspective of an author can become problematic because players act in the game’s story world rather than in the author’s head (Shaker et al., 2016). Procedural generation of stories achieved through simulation is a more straightforward approach for games because it has the advantage of talking about the exact same elements, places and events that the player will be interacting with in the game. On the other hand, the narrative arc and author-level goals that a human author provides are very useful. A hybrid system that combines planning of author-level goals on top of generated story-world events is thus highly desirable (Mateas & Stern, 2004; Riedl & Young, 2010).

In most computer games the scenarios, levels, storylines etc. are almost completely made in advance. Creating all this game content is a major effort and consumes a large part of the development budget (Yannakakis et al., 2009). In contrast to most major commercial publishers, Serious Game developers and/or studios seem to operate in a much smaller and more intimate environment. They do not have the budget required to afford professional game designers and the time it takes them to create an extensive amount of content. Instead, it is preferable to take a different approach: developing systems for automatically generating and adapting scenarios to given specifications and constraints. Instead of multiple days or months of extensive manual labour, complete environments can be created from algorithms that were given merely initial descriptions or even just random generator seeds (Yannakakis et al., 2009).

The Serious Games research area currently suffers from a lack of systems that have the ability of adapting storylines and narratives to the player without doing this at the cost of accessibility (Lopes, 2010). A direct consequence of this lack is Serious Games often ending up with generic, impersonal, limited and stereotyped training scenarios that prevent the true potential of the game to express itself (Lopes, 2010).

Another study also addresses the problem of the so-called “sameness” and lack of variety in gameplay (Charles et al., 2005). It proposes adaptive game technology as a solution, stating it is possible to dynamically tailor a game to individual players by using player modelling techniques and adaptive game technologies. Discussed in Player experience and adaptivity, the benefits of player adaptivity are numerous. The most important one being the improvement of player engagement, which is linked positively to desirable learning outcomes (Carini, Kuh and Klein, 2006) and thus ideal in a Serious Game situation.
To summarize the above content, the problem can be effectively broken down into two main elements:

- **Extensive manual labour** (Shaker et al., 2016; Yannakakis et al., 2009; McIntyre & Lapata, 2009)
- **Generic, impersonal, uninteresting Serious Games** (Lopes, 2010; Charles et al., 2005)

Which can be respectively solved by:

- **Procedural (narrative) generation** (Shaker et al., 2016; Mateas & Stern, 2004; Riedl & Young, 2010, Yannakakis et al., 2009)
- **Content adaptivity** (Lopes, 2010; Charles et al., 2005)

In an attempt to solve the above-mentioned elements of the problem by using the respective proposed solutions, some objectives for this thesis can be set up:

- Create an extendable system/tool that procedurally generates narratives.
- Implement existing techniques as modules for specific issues.
  - Create a module that extends stories based on existing knowledge.
  - Create a module that translates machine output into human language.
- Improve generated story quality by adapting the content to the player to increase player experience.
- Reduce manual labour in content creation for Serious Games.
- Experimentally evaluate the tools created.

Specific implementations on how these objectives are reached are further elaborated upon in the next section, **Method**.
4 Method

This thesis describes the development of a prototype for an interactive, modular and extendable tool that poses a solution to the earlier described problem of extensive manual labour (Shaker et al., 2016; Yannakakis et al., 2009; McIntyre & Lapata, 2009) and the problem concerning generic, impersonal and boring Serious Games (Lopes, 2010; Charles et al., 2005). The solution to those are procedural (narrative) generation and content adaptivity. The program aims to combine existing techniques, written into separate modules, into one central core application with an interactable and accessible user interface. This way, the tool helps bridge the gap between generated and human-authored stories (Mateas & Stern, 2004; Riedl & Young, 2010). After the implementation of the tool, its effectiveness is then evaluated in two separate experiments, both testing different modules.

4.1 Modularity

Due to the sheer size of both elements of the solution, procedural narrative generation and content adaptivity, this thesis limits its scope to providing a prototype for the tool and testing some modules that were developed and / or integrated in this thesis.

The main idea behind the tool is to set it up in a modular manner, so that new modules can always be added. This renders the tool permanently available for expansion in the future. This program is completely open-source and is available on GitHub, making the source code easily accessible for everyone and allowing smooth alteration and expansion upon the prototype. The tool is meant to be expanded by designers with specialized modules that are needed for specific games or situations.

Modularity is, from the definition proposed by Schilling (2000), a general systems concept. It is a continuum describing the degree to which a system’s components can be separated and recombined, and it refers both to the tightness of coupling between components and the degree of which “rules” of system architecture enable, or prohibit, the mixing and matching of components (Schilling, 2000). Derived from this definition, every system is in some way or another modular. However, when a system is referred to as a modular system, the degree to which the components can be separated is high. Different techniques and components used in this thesis will consequently be referred to as modules.

I chose this particular approach to the problem because of the accessibility and freedom it offers. An independent program is not tied to restrictions such as the game engine or the programming language the game is being developed with. This approach offers a wide range of possibilities as it appeals to the whole spectrum of Serious Game design. It can be used by designers that are making an open-world game where the story is the backbone to teach
people certain behaviours, but also for Serious Game designers that are developing a board game that doesn’t involve any technology in the game itself.

4.1.1 Ceptre module

The first module that extends the tool proposed in this thesis is an implementation of Ceptre (Martens, 2015). Ceptre is the core module for the narratives generated in the conducted experiments in this thesis.

In short, Ceptre is a language for modeling generative interactive systems. It is a rule specification language that has been created specifically to allow fast prototyping for experimental game mechanics with a focus on domains that rely on procedural narrative generation.

A comprehensive explanation of Ceptre is provided in the eponymous chapter.

4.1.2 Common sense module (ConceptNet & WordNet)

Commonsense knowledge was chosen as a potential module for the prototype presented in this thesis as a result of the potential shown through McIntyre's (2011) earlier studies proposing the incorporation of additional knowledge bases into the generation process of stories.

The concept of commonsense knowledge has already been thoroughly explained in the section commonsense knowledge. Summarized, commonsense knowledge is knowledge that consists of facts about the everyday world that are expected to be known by every ordinary person (Singh et al., 2002).

Commonsense knowledge is the second module that has been implemented in the tool proposed in this thesis. For this module, WordNetApi (Gerber, 2013) was adapted and ported to fit into the tool. Through the implementation of this module, the tool has access to all functionality of WordNet (Princeton University, 2010). WordNet is optimised for lexical categorisation and word-similarity determination while ConceptNet (Liu & Singh, 2004) is optimised for making practical context-based inferences over real-world texts and its dedication to contextual reasoning.

4.1.3 Language interpreter module

The third module that was implemented in the tool is a simple language interpreter module. This module is the first one that has been developed from scratch by the researcher. This module was written during the setup of the first study. The output from the Ceptre module of the tool, is unfortunately in a format that is not immediately readable for people that are
unfamiliar with it. In order to not introduce subjectivity, using a human author to transcribe the output was not a viable approach. To resolve this issue, a simple language interpreter module was written.

This interpreter was given the task of rendering the output log into a readable format, represented by simple, single sentences. Due to the simplicity of the module’s implementation, some drawbacks arise. For instance, the algorithm cannot create referring expressions or conjoin sentences that share a subject and/or object. These hindrances make the story sound very repetitive and give away that the story was not written by a person. An example demonstrating the difference in stories with and without referring expressions and conjoining of sentences can be found below.

From Story 3 of the questionnaire, the story without referring expressions nor conjoining of sentences:

The mom leaves the house and goes to the woods.
The mom wanders in the woods.
The mom gets lost in the woods.

The same story, with referring expressions and conjoining of sentences:

The mom leaves the house and goes to the woods.
The mom wanders in the woods and gets lost.

From the second version of the story it is clear that adding referring expressions and conjoining sentences is a huge improvement upon the first. Despite not having the ability to apply either of these, the simple language interpreter is still an enormous improvement compared to the raw output from the program, see figure 2 below and appendix C.
The logic behind the interpreter is simple, it uses regular expressions to parse the output and replaces the computer generated sentences accordingly. The full source code is available in the same repository as the final application (Declercq, 2018).

Due to the earlier explained drawbacks, for the second study a human author approach was eventually taken instead. An explanation and motivation can be found in the respective study chapter.

4.2 Evaluation of the tool

Evaluation of the tool is done through two studies executed in this thesis. By evaluating the results of the individual implemented modules, the usefulness (usability) of the modular tool can be measured. The content that is present in the experiment is exclusively set up by the designer using the toolbox.

4.3 Ethical considerations

During the conduction of both studies, the guidelines concerning the research ethical principles with human subjects set up by Vetenskapsrådet (2002) have been followed:

- The participant is informed about the overall research plan, aim of the research and methods used are communicated.
- The participant is informed about the fact that participation is completely voluntary and that they have the right to cease participation at any time.
- The data collected through these experiments is strictly used for the purpose of the conducted research. This information is in no way used for a commercial and/or non-research specific purpose.

Regarding the first part of the second study in particular, the privacy policy of Idrlabs (2018) explicitly states that the CT-NEO-IP-III test does not obtain personally identifiable information about participants.

Considering some stories presented in the second study contained possibly sensitive subjects such as depression and suicide, a small notice was stated in the introduction of the study. The message stressed one of the aforementioned guidelines: the fact that the participant can stop reading the stories at any time and is in no way obliged to complete the experiment.

In general, the fact that the data collection by the researcher in the following experiments is completely anonymous for both experiments means that no personally identifiable information is collected. This ensures all guidelines considering protection of personal information are followed as well.
4.4 First study: story generation, common sense and language interpretation

The first study in this thesis tests the general usability of the output this tool provides. As this is the first test run ever for the tool, even the most basic things have to be tested. The main goal of this study is to test whether Ceptre proves to be a valid choice as the core module for narrative generation. On top of that, the study evaluates if common sense knowledge can be used to extend stories and/or make up new rules. The previously discussed language interpretation module was written specifically for this study to transfer the information from the tool to the reader in a more readable manner, and will thus also receive its first evaluation moment through this study.

Further implementation, procedure and result details can be found in its own chapter.

4.5 Second study: adaptivity through the Ceptre module

The second study conducted in this thesis is designed to test the adaptivity that is natively present through the Ceptre module’s built-in ability to switch its non-determinism with input from the player or designer. This study consists of two components. The first component is a personality test based on the Big Five personality traits, also referred to as the five factor model (Murray & Mount, 1991). The second component of the study is the evaluation of four presented stories, similar to the first study. This time however, instead of having half of the stories made with commonsense knowledge rules, half of the stories were adapted to the results of the first component, the personality test. Each participant received one story that was chosen based on their highest scoring trait, another one was based on their lowest scoring trait and the remaining two stories were the same for all participants.

Further implementation, procedure and result details can be found in its own chapter.
5 Procedural narrative generation

5.1 Narrative adaptivity research

Frutos-Pascual and Zapirain (2015) have conducted a trend analysis concerning the employment of certain artificial intelligence algorithms that relate to the learning in the fields of Serious Games and decision making. In their paper, an explanation of their data analysis is included which divides the research of narratives for Serious Games into two main categories: decision making and machine learning.

As discussed in Player experience and adaptivity, Lopes (2012) is convinced the current lack of player adaptivity in Serious Games can be prevented by disallowing training modules from following rigid patterns. Multiple papers that have been compiled in the review presented by Frutos-Pascal and Zapirain (2015) present the collection of user data throughout the game as a possible solution. Often referred to as data mining, this technique constitutes an important field within player adaptivity. By utilising this mined data, the possibility of adapting the game narrative/scenario in real-time arises, which enhances the player process and subsequently the learning process.

Procedural generation of storylines is somewhat different from generation of other kinds of procedural content, because storylines are an unusual kind of content. They often intertwine pervasively with gameplay, and their role in a game can depend heavily on a game’s genre and mechanics.


To elucidate particularly the last part of above quote mentioning the dependence of a narrative’s role on a game’s genre and mechanics in the context of Serious Games, section Stories in games can be referred to. This section explains the possible roles of the narrative in (Serious) games in more detail. On the other hand, section Player experience and adaptivity sheds light upon the role of a Serious Games narrative towards player adaptivity and those correlations to the immediate learning for players, the ultimate goal of every Serious Game.

The greater part of modern Serious Games adaptivity research focuses on adapting behaviour rather than content (Lopes, 2010). Pierce et al. (2008) propose a system for adapting non-player characters behaviour to enable a personalized learning experience instead. The focus lying on adaptive behaviour rather than content means that instead of adapting the content itself, the adaptiveness is created through changes of other in-game elements. This difference can be easily demonstrated by an example taken from the open-world game Skyrim, a commercial game by world-renowned publisher Bethesda. The highest moments of immersion and most memorable situations that occur in franchises, such as The Elder Scrolls and Fallout, find their origin in the interchangeability and interactiveness that results from the inner relationships between their non-playing characters, frequently referred to as NPCs.
In *Skyrim*, the narrative behind the civil war between the *Imperial Legion* and the *Stormcloaks* is often told to the player by NPCs that play an important role in the main quest line. They are usually dull and easily forgettable, however in open-world games the player has the freedom to leave the main quest at any time and explore the vast amount of content the open world has to offer.

It is during one of these roaming sessions that I personally experienced the ongoing Civil War first-hand. I had accidentally lured a patrol of mounted *Imperial Soldiers* to a campfire surrounded by feasting *Stormcloaks*. The opposing parties spontaneously started a brawl. It was at this moment that I linked the events to the dull, uninteresting and unadapted stories that were told during the main quest line. The behaviour of the *Imperials* had adapted to me, the player, in the sense of them following me and had simultaneously intertwined with the already existing, scripted behaviour.

This adaptive behaviour was caused by *Skyrim*’s underlying disposition system that acts as a scale for NPC-friendliness towards the player. Different events and possibilities are triggered through different disposition levels. A high disposition value on an NPC renders the player multiple advantages. They can receive discounts in stores, random gifts or even completely novel storylines can be unlocked through having sufficient disposition. Low disposition on the other hand results in negative effects. Both positive and negative disposition is reflected in the speech the characters use towards the player as well.

All of the aforementioned adaptivity concerning *The Elder Scrolls* is created by adaptive behaviour rather than adaptive content. The gameplay of *Skyrim* combined with technical information of the underlying system is an excellent example how this adaptive behaviour works in the real world of game development.

### 5.2 Recurring elements

When studying current research and implementations of procedural narrative and storyline generation for computer games, a few recurring patterns and elements can be observed. Even though the names over different papers did not match, a clear pattern of 3 structures that are executed in the same sequence can be retrieved. Narrative generation finds its origin way before the actual game is implemented. During development of most games and Serious Games, the virtual game world is built and remains virtually invariable throughout the game. In order for adaptivity to be introduced, this approach needs to be altered. Different techniques that tackle this issue have already been established. Declarative optimization-based drama management, in short DODM (Nelson et al. 2006), and the rule specification language *Ceptre* (Martens, 2015) are two examples.

The initial state is the first structure that is commonly found in current implementations. It is often also referred to as context. This is the part of the process that sets the scene for the first act in previously mentioned rule specification language *Ceptre* for example. The engine itself is assigned the role of state manager, the so-called playwright. It has the task to keep
track of the possible rules and applications and has to determine which one of the next scenes is applicable, as several might be possible. In case of Ceptre the engine can either be taught to freely choose any random possible scene or it can be given other sources of input that determine which scene will be picked next. One such source is player input, more about the specific implementation and possibilities can be found in rule specification language Ceptre. In the initial state part, the role of the program designer consists of describing the means and requirements by which one state can possibly transition into another. The actual transition itself is handled by the state manager, that prepares the next scene by issuing cues to the involved in-game actors.

5.2.1 The initial state or context

The initial state mainly consists of specifications set by the designer. These include, but are not limited to descriptions of characters, locations, physical objects, possessions and sentiments between characters. This is similar to the aforementioned disposition system present in *The Elder Scrolls V: Skyrim*.

On top of these single element specifications, rules that consist of several combined elements are specified. They describe how the earlier specified elements can change over time and/or interact with each other. This conjunction of individual elements is what leads to possibly interesting logical developments over time. During this process of initial state set-up, the designer/author plays a very crucial role as he is the single source for all elements and rules.

Similar to the state manager from Ceptre, declarative optimization-based drama management, as described in Nelson et al. (2006), uses a drama manager. Its sole purpose is to reconfigure the game to the player, thus adapting it. The idea behind DODM is to avoid linearity in the plot structure, its main goal is to completely decouple the game mechanics from the plots, which leads to two different systems. One system handles the mechanics and the other handles the plot, the latter is the aforementioned drama manager.

Setting up DODM has three prerequisites, namely the provision of plot points, the provision of possible drama manager actions and an appropriate evaluation function. Whenever a previously specified plot point occurs during gameplay, the drama manager receives a request from the game to receive an immediately executable action. At this stage the game awaits a possible DM action as a response. This resulting drama manager action can be viewed as a direct intervention from the drama manager into the narrative. DM actions have a major impact on the plot as they either have a direct impact or they can render certain player actions permanently impossible, re-enable previously disabled ones, etc.
5.2.2 The progression or evolution

After the initial state has been set up, the progression or evolution stage is due. This is the phase where the specifications of the narrative change over time with regard to the previously defined rules. This is done by the drama or state manager, depending on how the specific implementation has named its managing entity. The state manager of Ceptre approaches this behaviour in a non-deterministic way, details about the benefits of a non-deterministic procedure are explained thoroughly in rule specification language Ceptre.

In the declarative optimisation-based drama management implementation, described in Nelson et al. (2006), a game’s narrative is seen as a sequence of plot points. When thought appropriate, the drama manager can employ its authority to intervene in the story using its aforementioned actions. DM actions are evaluated through a dedicated evaluation function. Stories that score high using the evaluation function are chosen in order to maximize the potential narrative quality. An issue native to DODM that is also addressed by Nelson et al. (2006) is the required level of specification of the aforementioned plot points’ corresponding DM actions. At the lowest level, a single line of code could theoretically directly trigger a DM action. This raises concerns towards expandability, as every individual trigger possibly needs tweaking through future playtests if it is considered too likely or vice versa. Enabling higher level specifications, and consequently solving the problem, can be done by employing a recognizer. A recognizer emits a signal when it detects a specified event happening. On the other side of the communication, this emitted signal is subsequently picked up by a refiner. The refiner’s job is translating the received abstract DM actions into concrete activities that are directly applicable to the game world.

5.2.3 The debriefing or evaluation

The final persistently recurring concept is the evaluation, often also referred to as debriefing or assessment. As is the case with the previous two recurring concepts, the specific implementation of the concept varies across different research. However, a few implementations are more prominent than others. Most current Serious Games related research concerning adaptivity evaluation still centers its focus on, sometimes computer-enhanced, debriefing/review sessions that take place after the action (Raybourn, 2007). The conclusions from this approach are based around data that has been collected during the play session. They act as support for feedback sessions that involve the player themselves and an instructor who is responsible for translating the assessment data into information that is of incredible value for current artificial intelligence methods aimed towards improving player experience.

Another noteworthy approach of assessment that does not involve any human intervention is computation of proof search, as proposed by Hodas and Miller (1994). This approach handles a sequence as input and attempts to create a proof as output. If no proof can be
found, it declares failure. Martens (2015) describes two different implementations of proof search: the backward chaining or goal-driven proof search and the forward chaining or assumption-driven proof search. Ceptre implements the latter. This is what allows Ceptre to model the simulations and allows the production of new information through simulating rules on currently known facts, without the need for a specific goal. Choosing which rules to apply is in direct correlation with the general game progression and thus the general expansion of the involved narrative.

As mentioned before, DODM uses an evaluation function as its debriefing element. The author of this evaluation function is the solely responsible for specifying what exactly constitutes a good story, not the effectively used elements that determine the conveyance of such story. This presents an optimization problem as DODM tries maximizing the player experience through a well adapted narrative. Nelson et al. (2016) implements reinforcement learning as an approach to solve this.

5.2.4 Conclusion

Through this existing research on procedural narrative generation and an analysis of recurring elements, it can be concluded that a good base for a workflow when generating narratives procedurally is starting from an initial state that is carefully manufactured by a human designer/instructor. A managing entity should be in place that can be called upon later during gameplay, or the process of generation in case of a non-adaptable game scenario, to guide the storyline into the direction that offers an as optimal as possible player experience. This managing entity can ideally be used for different sorts of games once it has been shaped. The last crucial step for procedural narrative generation is a good evaluation function that is able to assess the quality of the manager’s performance. This data can then be used to improve the manager for future executions.

A compact visual representation of the commonly recurring elements in procedural narrative generation and their sequence can be found in figure 3.

![Figure 3](https://via.placeholder.com/150.png "Recurring elements in PNG")
6 Ceptre, a rule specification language

The application described in this thesis integrates Ceptre, a language for modeling generative interactive systems (Martens, 2015), as the main narrative generation module. Ceptre is a rule specification language created to allow fast prototyping for experimental game mechanics, especially in domains that rely on procedural generation.

Ceptre improves upon the Celf language (Schack-Nielsen and Schürmann, 2008) with two notable features. Firstly, it allows for interactivity and secondly it introduces stages. The interactivity that is natively implemented into the language offers a wide range of possibilities when working towards possible content adaptivity. Ceptre handles the evolution (phase 2, specified in procedural narrative generation) of the context and the rules without specific rule ordering. Any rules that were specified in the Ceptre script are evaluated for application against a subset of the current state configuration. If possible, the result of the operation is the same subset with the non-constant members replaced with the consequences of the rule. This process leads to causal structures. A causal structure occurs when a rule produces resources and another rule consumes a subset of these resources. In this case the rules are in a causal (= dependent) relationship with each other (Martens, 2015).

6.1 Non-determinism

The fact that there is no fixed order in which rules are evaluated for use and that several may apply at any given moment, renders the behaviour of Ceptre non-deterministic. In a non-deterministic environment, any input given can lead to varying output. In short, non-determinism means that one input does not necessarily lead to another outcome, but instead to multiple, different outputs. The exact output that is produced, varies across different runs of the program. The presence of this type of behaviour in the evolution stage of procedural narrative generation is a great advantage. Through non-determinism, a relatively small amount of input can lead to a lot of variety in generated content.

On top of this, the language offers the designer the possibility to exchange this non-determinism with input from an external source. The most obvious choice in this case is to allow the player, in case of real-time generation, or the designer himself, in case of setting up the universe and designing the experience, to provide input to the manager. In case of the former, content and/or rules can be guided to an experience or storyline that is more close to what the player likes and thus provides a better learning experience. This is a direct implementation of what has earlier been described in player experience and adaptivity as player adaptivity.

At any given time the interaction can be swapped back to the non-deterministic behaviour of the manager without input. Having the freedom to switch between these two possibilities

24
gives the designer a lot of options. For example, the designer can give the player the freedom to decide what happens on crucial points in the storyline and thereafter hand the control back to the non-deterministic behaviour default of Ceptre. Even though this was a single decision by the player, it potentially has a massive impact on the further course of the story and can change the remaining flow of the story and outcome completely.

6.2 Debugging

Every program execution of Ceptre is recorded in a trace that keeps track of the dependency information of the causal structure. This trace is outputted in two distinct forms: once in the form of a text file named log.txt and once as a .dot file. These resources represent the, what could be called, thought process of the program. It contains information about the steps that were taken in the evolution phase of the generation. Both of these resources make for excellent debugging tools, as they not just contain a linear record of which rules were applied, but a structured term instead. This can be presented as a directed, acyclic graph that maps rule applications onto their causal relationship (Martens, 2015). This information stored in the aforementioned .dot file can be visualised using an auxiliary graphical visualisation tool such as GraphViz (Ellson et al., 2001.), an example is shown below in figure 4.

![Figure 4](image-url)
Next to being a great debugging tool, these resources provide a viable source of information for a Serious Game designer because they allow insight in feedback loops of game mechanics. When unifying all instances of a rule being fired in a trace, the designer can easily deduct which rules consistently come before others and what influence they have on other specific rules, both extremely useful aspects when designing games (Martens, 2015). Both the graph and the log file are irreplaceable as sources when, manually or not, evaluating the rules and initial states that were generated. It is one of the most direct and only ways of evaluating the direct consequences of rules, which is important when deciding to either add or omit a rule for the final generation script.
7 Study: Story generation, common sense and language interpretation

7.1 Implementation

Ceptre (Martens, 2015), as described in Ceptre a rule specification language, was chosen as the core module for the generation of this study because of its non-determinism. As explained in previous sections, non-determinism allows multiple different outputs to easily be generated from any given input. Even though not all output might be immediately usable, the time required for the designer to manually evaluate the output quality is negligible when compared to the time it would have taken for the designer to come up with a similar result.

Another major advantage that was taken into consideration when deciding which approach to use, is that Ceptre outputs a trace. All program execution done through Ceptre is stored in the trace, which keeps track of the rule dependency information of the causal structure. The trace is outputted in two distinct forms, one being a text file (named by log.txt by default) and the other being a .dot file.

These resources represent the so-called thought process of the program, information about all steps that were taken during the evolution phase of the generation are stored. The importance of this information for debugging purposes can hardly be understated. They are invaluable as a debugging tool since they don't contain just a linear record of which rules were applied, but a structured term instead. The information that is stored in the aforementioned .dot file can be visualised using an auxiliary graphical visualisation tool such as GraphViz (Ellson et al., 2001.). An example of such visualised trace is seen in figure 4.

The common sense knowledge database that was used for this study is ConceptNet (Speer & Havasi, 2012). As earlier described in the chapter about commonsense knowledge, ConceptNet is a database that consists of common sense data that has been automatically extracted from human authored sentences that were collected as part of the Open Mind Common sense project (Singh & Barry, 2003). It provides commonly known relationships between certain objects. As an example, table 1 presents the information that is retrieved from ConceptNet with the word wolf provided as input. ConceptNet also provides a visualisation of this data, available at http://conceptnet.io/c/en/wolf.
Table 1  

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Examples (where x = wolf)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types of x</td>
<td>coyote, white wolf</td>
</tr>
<tr>
<td>x is a type of</td>
<td>canine, animal</td>
</tr>
<tr>
<td>x is capable of</td>
<td>give a howl, smell a human</td>
</tr>
<tr>
<td>location of x</td>
<td>a forest</td>
</tr>
<tr>
<td>x has</td>
<td>a reputation of being sneaky</td>
</tr>
<tr>
<td>x is defined as</td>
<td>ancestor of all dogs</td>
</tr>
<tr>
<td>antonym of x</td>
<td>sheep</td>
</tr>
</tbody>
</table>

The story of Little Red Riding Hood in a simplified English version (British Council, 2017) was used as the core content of the experiment. This story and version was chosen specifically because of its length and difficulty. Because this story is so short and easily understandable, it is very extendable and thus renders itself as an ideal source when attempting to generate new stories.

The rules that are used for the generation of the presented stories have been manually extracted from the story by the researcher. Some examples can be found in figure 5. For the full rules, refer to appendix A and appendix B.
The story that is generated is retrieved from the log file. However, the output is not directly readable for people that are not familiar with the format. Seeing a story in this format possibly has a negative influence on the scoring evaluation system. Ideally, this data from the output would be written into a story format by a human author instead. This is however not a plausible approach for this particular study as it might introduce subjectivity. There is no guarantee that a human author will interpret the same output in the same manner every time.

A new module was written to resolve this issue, the simple language interpreter module. The module has earlier been thoroughly explained in the Method section.

### 7.2 Procedure

Each participant of the study is provided 10 short stories, generated using the aforementioned techniques. Half of the given stories are generated without the addition of common sense rules (appendix A), while the other half is generated with common sense rules included (appendix B). The participants are deliberately not informed of this fact to ensure the grading of the stories happens in an unbiased manner. The sequence in which the stories are presented is randomized before the start of the experiment, ensuring the sequence of the presentation has no influence on the received score. The full short stories that were provided to the participants can be found in appendix D.

A total of 29 people participated in the study. Ages range from 18 to 56 with most participants being within the age range of [20-25]. People with different levels of education were selected to avoid bias. All participants are acquaintances or friends of the researcher and were personally contacted by him.

### 7.3 Presentation

The study is presented to the participants through Google Forms. Considering the participants have varying backgrounds and familiarity with terms like procedural generation, the introduction to the experiment provides some clarification. To avoid problems with terms that might cause possible confusion when taken out of context, some characters are clarified in the introduction as well. For example, to the program the character called “mom” is just a character. To the reader however, it is important to know who’s mother this character is. Even if this might be easily deductible, the explanation is provided to exclude confusion.

Each participant is provided with the following instructions before starting the questionnaire. As soon as the participant starts the questionnaire, the researcher provides no input except for these instructions.
This experiment is part of a study concerning procedural storyline generation, or explained simply: computers that write stories. The study is currently in development so any feedback you can provide through assessing the short stories in this questionnaire will be very helpful for future development.

In this experiment you will be asked to judge a set of computer-generated short stories. They have been provided to you in a readable form, but this is by no means the final product. The rendering of generated information into sentences, also known as Natural Language Processing is beyond the scope of the project. Therefore, you are kindly asked to not judge based on the writing style.

Clarification on the characters:
Mom = the girl’s mom
Granny = the girl’s granny

You will be asked to rate the stories on a scale of 1 to 5, base on the following factors:

**Plot**: Rate the story from little to no plot (1) to very strong plot (5).
The plot of a story is the integration of cause and effect, a strong plot means the events progress in a logical matter and make sense to happen in sequence.

**Interest**: Rate the story from very boring (1) to very exciting (5).

**Coherence**: Rate the story from completely unorganized (1) to very coherent (5).
This is not the same as plot. Things can fit well together without having a cause and effect relationship.

**Believability of the characters**: Rate from hard to believe (1) to very believable (5).

Please don’t rate the stories based on the writing style.

### 7.4 Evaluation method

The method used for the story evaluation is an adaptation from a discussed evaluation method by McIntyre (2011). The participant is asked to rate the stories presented to them based on four different criteria on a scale of 1 to 5, where 1 is the lowest possible score and 5 the highest. The distinct criteria for evaluation are as follows: plot, interest, coherence and believability of the characters. Individual descriptions for each criterium are also provided to ease the understanding of the respective scales for the participant.

The participants are explicitly asked not to rate the stories based on the writing style, considering that is beyond the scope of this experiment, as has been thoroughly explained in Implementation. Next to these ratings that approach the research in a quantitative way, two open questions are posed to the participant for each story.
Question 1: Please write briefly whether or not you enjoyed the story and why.

Question 2: Do you have any remarks on extremely good or extremely bad elements?

Both of these questions could be very useful if the study was approached from a qualititative perspective. However, due to the questions being optional, there was insufficient data. Therefore, apart from some exceptional remarks discussed in the next section, the results of this study have been achieved through a quantitative approach.

7.5 Results and discussion

The results of the experiment were retrieved from Google Forms and manually put into a spreadsheet by the researcher. The spreadsheet is available at: https://goo.gl/qMFNKZ.

As can be deducted from the spreadsheet, a story score is calculated for every participant on every individual story. This story score is the mean of the scores given by the participant on the individual criteria, providing an image of how well the participants perceived said story. For example, if the participant gave following ratings: 4 for plot, 2 for interest, 3 for coherence and 3 for believability of the characters then a story score of 3 is calculated because \((4 + 2 + 3 + 3) / 4 = 3\).

An unpaired t-test was carried out on the gathered result. Having 28 participants and 5 stories per category leads to a sample size of 140 per category. The T-test was done using the story scores described earlier.

The result of the unpaired t-test showed a t value = 0.4112 with degrees of freedom = 275 and a (two-tailed) P value = 0.618. These results are far from being statistically significant.

With the results of the unpaired t-test being not statistically significant, it can be concluded that adding rules retrieved through common sense logic does not improve story quality. However, this on the other hand shows that adding these rules does not make stories worse either, which in its turn confirms that common sense rules can be used to flesh out a story.

It is also interesting to see that the story with the highest final score (3.71 out of 5) is generated with the addition of common sense rules, while the story with the lowest final story score (2.48 out of 5) was generated without. This might indicate that stories using common sense rules might have more potential than stories without.

Another possibly noteworthy observation is the fact that results about certain stories vary greatly across participants. For example, story 2 (no common sense rules) has a minimum
score of 1.75 and a maximum score of 4.5 with a mean score of 3.1. The participant that gave the lowest score answered to open question 2: “I did not enjoy the story, there was no start point, main story and end point”, while the participant giving the highest score stated “Enjoyed it because it made sense.” In order for these scores to be interpreted usefully, a qualitative study should be conducted that goes deeper than the above mentioned superficial short comments. As mentioned earlier, the optionality, short nature and superficiality of the provided answers to the open questions are the main reason this study is conducted employing a quantitative approach rather than a qualitative one.
8 Study: Adaptivity through the Ceptre module

8.1 Participant feedback

After the first study conducted in this thesis, the researcher had a small, informal talk with each of the participants to thank them for their participation and ask them in a friendly environment what they thought of the test. None of these conversations were written down as they were not a part of the actual study.

However, these small informal feedback sessions gave some interesting insights about the procedure of the study itself rather than the content of it. Most people agreed on the fact that there were too many stories presented. As thoroughly explained in the procedure section of the first experiment, each participant was provided with ten stories in total of which five were generated with the addition of common sense rules and five without. According to the feedback this was a little bit over the top and made the experiment too long and monotonous.

Another thing pointed out by a large amount of the participants of the first experiment was the language in which the stories were presented. Even though it was stated multiple times in the introduction to the experiment that participants should not judge the stories based on writing quality, because natural language processing is a huge field and completely out of scope of this thesis. Explained in the Implementation of study 1, the generated stories outputted by Ceptre, regardless of whether common sense rules were used, are in an unreadable format for human readers. Knowing advanced natural language processing is out of scope, this problem was solved by writing the simple language interpreter module, as seen in Language interpreter module and appendix C.

Nevertheless, the story presentation and writing apparently still remained a hurdle to overcome for people that wanted to immerse themselves in the stories. In order to solve this particular problem, the language interpreter module was not used for the second, instead the output was manually transcribed by the researcher under a fixed set of simple rules. Through this approach, referring expressions and conjoined sentences are present in the presentation of the stories to the participants.
8.2 Procedure

8.2.1 First component
In order to adapt content to the player, information about the specific individual has to be known. Ideally, and always present in a real-world scenario, this data is gathered through the first few playthroughs of the (Serious) game for which content is being created. However, for this experiment there was no game in place and it would be unreasonable scope-wise to implement a game that can mimic the complexity of a fully developed game just for this specific experiment.

Therefore, as a compromise, the participants are presented with a personality test that assesses a personal score for each of the Big Five personality traits. The data extracted from this test will be used to adapt the content.

The Big Five personality traits, also referred to as the five factor model or in short FFM, is a classification for personality traits based on common language descriptors (Murray & Mount, 1991). The five factors are:

- Openness to experience
- Conscientiousness
- Extraversion
- Agreeableness
- Neuroticism

The specific test presented to the participants in this study is CT-NEO-IP-III (Idrlabs, 2018) and is based on the work of, among others, Gerber et al., 2010.

Underneath above mentioned global factors, a number of correlated more specific factors reside (Matthews et al., 2003). They are explained more in depth later in this study section.

8.2.2 Second component
The tool still uses the Ceptre (Martens, 2015) module for the narrative generation in this experiment because it has proven its worth during the first experiment. It was chosen as the core module mainly because of its non-determinism, through this it can generate enormous amounts of content with minimal input in minimal time.

Ceptre has the built-in ability to swap out its native non-deterministic behaviour with an external source such as player or designer input, also described in the Ceptre-specific chapter. In the case of this study, the researcher who fulfilled the role of designer was the external source that provided input. Through the unique custom implementation of the Ceptre module into the toolbox proposed in this thesis, also shown in figure 6 below, the
researcher used the user interface to adapt the stories to the results of the personality test (see also next section story selection).

**Figure 6** Screenshot from the implemented interactivity in the tool

The story of Romeo and Juliet in a simplified English version (British Council, 2017) was used as the core content of the experiment. Additional components such as excessive drinking possibilities and depression were added upon this core story by the researcher to mimic behavior suitable for other personality results.

In the second component of the study a total of four stories are presented to the participant, a lower number than the first experiment was deliberately chosen with consideration for the feedback provided by the participants of the first study, see participant feedback. Two of which are adapted to the personality test which is more thoroughly explained in the next section concerning story selection. The other two stories are the same for every participant, they form the control group. All stories presented to participants can be found in appendix E.

### 8.2.3 Story selection

Based on the aforementioned underlying factors of the big five personality traits, different stories are presented to different participants based on their results of the CT-NEO-IP-III personality test. A compact overview of the selection process is described below.

Participants that scored conscientiousness as their highest trait, of which an underlying factor is dutifulness (Toegel & Barsoux, 2012), were presented with a story where one of the protagonists dutifully defends his friend in a fight. Contrarily, people with conscientiousness as their lowest trait, received a story where one of the protagonists shows signs of low self discipline and sloppiness (Toegel & Barsoux, 2012).
Participants with their highest trait openness to experience, of which an underlying factor is their likeliness of engaging in risky behaviour or drug taking such as alcohol abuse (Ambridge, 2014), were presented with a story that involved excessive drinking. If openness to experience happened to be their lowest trait, they were presented with a story where the protagonist debates on staying home instead of going to a party.

Extraversion as a highest trait lead to stories with excessive drinking and partying as well. On the other hand, participants with a low extraversion were presented a story where the friends would go to the party, but they wouldn’t join themselves. A low extraversion level leads to introverted people that act more shy and reserved (Friedman, 2016) and are thus less likely to join a party.

In case of agreeableness scoring the highest, participants were presented with a story containing elements that show compassion and/or friendliness (Toegel & Barsoux, 2012). Some examples of these elements taken straight from the story that was actually presented for high agreeableness are: reconsidering fights, introducing friends to other friends that were not acquainted before and defending new friends. Stories presented to people with low agreeableness contained lots of fighting instead.

Last but not least, the neuroticism trait was considered. In case of this particular study this can be considered the most influential trait. Participants that scored neuroticism as their highest trait received stories that contained elements such as depression and possible suicide as a consequence, since Dwan & Onsworth (2017) describe that individuals with higher levels of neuroticism tend to have worse psychological well being. Friedman (2016) adds that they are also more prone to psychological stress.

8.3 Evaluation method

The evaluation method chosen for this study is the same as the one used in the first study, namely an adaptation from a discussed evaluation method by McIntyre (2011). The participant is asked to rate the stories presented to them based on four different criteria on a scale of 1 to 5, where 1 is the lowest possible score and 5 the highest. The distinct criteria for evaluation are as follows: plot, interest, coherence and believability of the characters. Individual descriptions for each criterium are also provided to ease the understanding of the respective scales for the participant.

This time however, considering feedback from the first study, the open questions have been omitted. These have proven to not be very useful if given on the side, and the optionality of them did not help in that aspect either. Most people that completed them did so out of politeness rather than interest.

36
8.4 Results and discussion

The results of the experiment were retrieved from Google Forms. Due to the nature of the branching in questions, because of the different personalities leading to a cluttered result sheet, the spreadsheet was manually edited and cleaned by the researcher to improve readability and usefulness. The cleaned spreadsheet is available at: https://goo.gl/34aokS.

Identically to the procedure of the first experiment, a story score is calculated on every individual story per participant. The story score is the mean of the scores given by the participant on the individual criteria.

The null hypothesis is that there is no difference in story scores for stories that have been adapted to the results of the personality test taken by the participants than story scores for the stories that have not been adapted.

The alternative hypothesis is that the story scores for stories that have been adapted to the results of the personality test taken by the participants are better than story scores for the stories that have not been adapted. An unpaired t-test was carried out on the gathered result. Having exactly 60 participants and 2 stories per category leads to a sample size n of 120 per category. The unpaired t-test was done using the previously described story scores.

The results of the unpaired t-test yielded a t value = 0.722 with degrees of freedom = 238 and a (two-tailed) P value = 0.471. These results are yet again not statistically significant. These results fail to reject the null hypothesis.

![Boxplot of second study results](image)

Figure 7    Boxplot of second study results
The results of the second study have been graphically represented through boxplots in the figure above. Here can be seen that both the adapted category and the original category are very similar. A few small but insignificant changes can be noted. It can be seen that the minimum for the adapted category is slightly better than the original category with a story score of 1.25 and 1.0 respectively. Another small positive change is that the first quartile of the adapted stories lies a little higher than the first quartile of the original, proving a small but insignificant improvement. Unfortunately, the same can be said for the third quartile, but in the other direction. The third quartile for the adapted category can be found lower than the third quartile of the original, which means a slight deterioration.

With the results of the unpaired t-test not being statistically significant, it can be concluded that adaptation of the story to the player/reader does not have an impact on the story score. Thus the adaptivity provided within the current implementation of the Ceptre module, combined with a personality test to tailor to does not yield better stories as a result in this case.
9 Conclusions

9.1 Summary
Two separate studies were conducted in this thesis. The first study was designed to test the general usability of the tool and the potential of Ceptre as the core module for narrative generation. On top of this, the experiment tested if rules composed of common sense knowledge from the ConceptNet database could be used for content creation. It also tested, to a lesser extent, the usability of the language interpreter module that was written specifically for this first study.

Ceptre proved, despite having to use quite a few workarounds in the workflow when coding in the language designed for Ceptre, to be a viable module for the narrative generation part of the tool. The results of the unpaired t-test conducted in this first study were statistically insignificant. This teaches us that adding rules retrieved through common sense logic does not improve story quality. However, it can be argued that adding these rules does not make stories worse either, which in its turn confirms that common sense rules can be used to flesh out a story. The third and last aspect evaluated in the first study was the usability of the language interpreter module. Even though the output by the language interpreter is a huge improvement upon the original output from the Ceptre module, direct feedback from the participants has proven it to be insufficient and hindering to the story.

The second study conducted in this thesis was designed to measure the improvements in story score when comparing stories adapted to the player, participant in our case, to story scores for stories that were not adapted to the player. The results of the unpaired t-test conducted in this study were yet again not statistically significant and they failed to reject the null-hypothesis. However, same case as the first study, the story scores did not decrease either.

9.2 Discussion
The tool proposed in this thesis is meant to be used as a go-to tool for Serious Game designers and creators for content creation. During both studies conducted in this thesis, the researcher took the role of game designer while making the stories that were presented to the participants. This means that the best judge of specifically the accessibility and usability of the tool is the researcher, rather than the results. The results give information about the usefulness rather than the usability.

During the setup of the experiments, the visual interface provided by the tool was very helpful. Being able to just have everything in one place rather than scattered had a positive effect on productivity. The integration of the Ceptre module for the generation in the tool was a lot more user friendly than the original console command controlled program.
The combination of existing techniques has proven to be very useful when creating new content. Whenever the researcher was out of inspiration, both Ceptre and the common sense integration provided useful new content, ideas and approaches. Not all ideas were usable as they were given, but the sheer speed at which the generator provided content made up for it. The tool does not restrict you in use of content, it can be used and / or reinterpreted by the designers in whichever way they like.

9.2.1 Positive remarks

+ The combination of existing techniques proved to be very useful for new idea and content generation.

+ Once a new module is set up and integrated, it works seamlessly in the tool itself and can be used immediately in conjunction with the other already integrated modules.

9.2.2 Negative remarks

- The implementation of the different modules into the tool was a little more cumbersome than expected. The time it takes might be a hurdle for future development, since the purpose of this tool is to save developer time.

- From personal experience, the researcher notes that the use of a combination of existing techniques sometimes turned out to work better than the actual combination of them in the tool. Just using them alongside each other was occasionally easier than having to implement and integrate them in the tool environment before being able to use them.

9.2.3 Research objectives

Most, if not all, research objectives and whether or not they have been achieved have already been explained throughout the thesis and the conducted experiments. This subsection presents a concise summary of the already established observations.

From the conducted experiments, it can be derived that the first research objective, creating an extendable system/tool that procedurally generates narratives, has been achieved. For both experiments, the researcher was able to use the tool to procedurally generate narratives that were presented to the participant.

The first part of the second research objective, creating a module that extends stories based on existing knowledge, was tested in the first study. The discussed common sense module that extends existing stories using existing knowledge was evaluated in the first study. The results of the unpaired t-test conducted were not statistically significant, stating that adding rules retrieved through common sense knowledge did not improve story quality. It was already concluded that this also proves that adding rules generated by the common sense
module did not make a story worse either and thus the objective of extending stories based on existing knowledge can be considered achieved as well.

The second part of the second research objective, creating a module that translates machine output into human language, concerns the language interpreter module. As already mentioned in the module-specific chapter, this module had some drawbacks such as not handling referring expressions nor conjoining sentences. Because of these drawbacks the second study did not employ this module. The research objective of creating a module that translates machine output into human language is thus only half-fulfilled by this thesis. For this reason, the future work section proposes a natural language processing addition.

The third research objective, improving generated story quality by adapting the content to the player to increase player experience, was tested in the second study. Results of this study proved statistically insignificant and thus the objective of improving story quality by adaptation was not achieved. However, the earlier described compromise of using a personality test rather than real game data might give a skewed view of the usability of the current module. Ideally a third study should be conducted that tests the current adaptivity implementation with real collected data from a game.

Through the previously discussed elements in this discussion section, the last objective, reducing manual labour in content creation for Serious Games, can be considered achieved. Once a new module has been integrated, an enormous amount of manual labour that would have been required without this tool is made obsolete and the process of actually creating new content is quick and does not require extensive manual involvement.

### 9.3 Future Work

From the very beginning, the idea behind the toolbox was to have it set up in a modular and expandable way. It is thus expected for the future work section to contain some proposals for future modules that can be attached to the provided prototype for the tool.

A much needed module for future research would be a decent natural language processing module. The drawbacks and issues that the language interpreter module brought forth have proven how useful it would be to have a decent nlp module. It would improve the output of the toolbox a lot as well.

There is another downside to the toolbox as well. When making a digital game it takes extra work to convert the output data of the provided program into the specific format that a certain environment requires. If for example a text handling a dog is provided to the system, the system is likely to generate behaviours for the dog such as barking, scratching, etc.
Next to generating behaviours, it is possible that a list of plausible moods are generated for an entity such as a dog e.g. angry, happy, anxious, surprised. With these moods, Ceptre rules like

$$\text{dog} * \text{unknown_object} \ -o \ \text{dog_surprised}$$

are likely to occur. These behaviours and moods are generated by processing data retrieved from commonsense knowledge databases such as ConceptNet (Speer and Havasi, 2012). If the designer wants to use these rules in a game engine, they have to be written in a form the program understands.

For example when using Unity, dog could be turned into a C# class that contains an enumeration with options for the different moods. An example of the possible code is shown in figure 8.

```csharp
public class Dog
{
    public enum Mood
    {
        Angry,
        Happy,
        Anxious,
        Surprised
    }
}
```

**Figure 8**  C# generated states to code example

This process however, is specific to the environment the game is being created in. It is impossible for the toolbox to provide an output format or code generation method for every possible engine and/or programming language. If the designer decides that automatic code generation from the output of the application would help the workflow, a specific generator script can be written for this case as an extension to the toolbox. It would be nice to get a collection of specific export modules into the toolbox, for example a Unity class generator.
10 References


10 Appendices

A - First study Ceptre code without the addition of common sense knowledge

% Types
character : type.
location : type.

% Predicates
house character location : pred.
at character location : pred.
visit character character : pred.
meet character character : pred.
dead character : pred.
fightToken character location : pred.
live character location : pred.

houseToken : pred.
livedHappilyEverAfter : pred.
endstageToken : pred.
runAway : pred.
scream : pred.
interrogate : pred.
hasntVisited : pred.
wolfNotEncountered : pred.

% Characters
mom : character.
wolf : character.
granny : character.
redhood : character.
woodcutter : character.

% Locations
spawn : location.
woods : location.
cabin : location.
granny_house : location.
mom_house : location.

% Stages
stage spawnStage = {
    % Spawn at a random location
    spawn_woods : at C spawn -o at C woods.
    spawn_cabin : at C spawn -o at C cabin.
}

stage mainStage = {
    % If the wolf meets redhood, he knows she is heading for grannies house
    wolfEncounter : $hasntVisited * wolfNotEncountered * at wolf woods * $at redhood woods -o at wolf granny_house.
% Any character can meet any character if they are in the same place
meet : $\text{hasn'tVisited} \ast \text{at } C \text{ woods} \ast \text{at } C' \text{ woods} - o

% If the wolf is in the woods, he has the possibility to eat anyone that is also in the woods
eat : $\text{hasn'tVisited} \ast \text{at wolf woods} \ast \text{at } C \text{ woods} - o \text{ dead } C.

% Anyone that is in the woods has the possibility to fight the wolf
fightWolf : $\text{hasn'tVisited} \ast \text{at } C \text{ woods} \ast \text{at wolf woods} - o \text{ fightToken } C \text{ woods}.

% Possibility to get lost in the woods
gets_lost : $\text{hasn'tVisited} \ast \text{at } C \text{ woods} - o.

% Possibility to wander around in the woods
wander_in_woods : $\text{hasn'tVisited} \ast \text{at } C \text{ woods} - o \text{ at } C \text{ woods}.

}

stage fightStage = {
    wolf_wins : fightToken C L - o at wolf L * dead C.
    character_wins : fightToken C L - o at C L * dead wolf.
}

stage houseStage = {
    % Consume the visit token and live happily ever after, no danger from the wolf
    mom_house : visit redhood mom - o livedHappilyEverAfter.

    % Visit grandma when wolf doesn't know about whereabouts bc he hasn't encountered redhood
    Grandma_no_wolf: visit redhood granny * wolfNotEncountered - o LivedHappilyEverAfter.

    % Consume the visit token and scream
    scream : visit redhood granny * at wolf granny_house - o scream.

    % Consume the visit token and redhoods location, run away to the woods
    runAway : visit redhood granny * at wolf granny_house * at redhood L - o at redhood woods.

    % Woodcutter heard scream
    woodcutterAlerted : scream * at woodcutter woods - o at woodcutter granny_house *
    fightToken woodcutter granny_house.

    % Woodcutter didn't hear scream, because he was in the cabin
    woodcutterNotalerted: scream * at woodcutter cabin - o dead redhood.
}

% Stage transferring rules
qui * stage spawnStage - o stage mainStage.

qui * stage mainStage * $\text{fightToken} C L - o stage fightStage.
qui * stage fightStage - o stage mainStage.

qui * stage mainStage * houseToken - o stage houseStage.
qui * stage houseStage * $\text{fightToken} C L - o stage fightStage.
% Contexts
context init =
{at redhood woods, at wolf woods, at woodcutter spawn, house granny granny_house, house mom mom_house, hasntVisited, wolfNotEncountered}.

#trace _ spawnStage init.
B - First study Ceptre code with the addition of common sense knowledge

% Types
character : type.
location : type.

% Predicates
at character location : pred.
visit character character : pred.
meet character character : pred.
dead character : pred.
fightToken character location : pred.
live character location : pred.

visitToken : pred.
livedHappilyEverAfter : pred.
endstageToken : pred.
runAway : pred.
scream : pred.
interrogate : pred.
hasntVisited : pred.
wolfNotEncountered : pred.

% Characters
mom : character.

wolf : character.
granny : character.
redhood : character.
woodcutter : character.

% Locations
spawn : location.
woods : location.
cabin : location.
granny_house : location.

% CONCEPTNET
smell character character : pred.
smellToken : pred.

howl character character : pred.
howlToken : pred.

cutTree character : pred.
cutTreeToken : pred.

offerCookies character character : pred.
cookieToken : pred.

cookTurnips character : pred.
turnipToken : pred.
% Stages
stage spawnStage = {
  % Spawn at a random location
  spawn_woods : at C spawn -o at C woods.
  spawn_cabin : at C spawn -o at C cabin.
}

stage mainStage = {
  % The wolf can howl at a character if he is in the woods
  howl : $hasntVisited * $at wolf woods * $at C woods * howlToken -o howl wolf C.

  % The wolf can smell a human
  smell : $hasnVisited * $at wolf L * $at C L * smellToken -o smell wolf C.

  % If the woodcutter is in the woods, he can cut a tree
  cutTree : $hasnVisited * $at woodcutter woods * cutTreeToken -o cutTree woodcutter.

  % If someone visits granny, she can offer cookies
  offerCookies : $visit C granny * cookieToken -o offerCookies granny C.

  % If cookies were offered, they can be eaten
  eatCookies : offerCookies C C' -o.

  % Granny can cook turnips
  cookTurnips : turnipToken -o cookTurnips granny.

  % Leave the house and go to the woods
  leaveHouseToWoods : live C woods -o at C woods.

  % If the wolf meets redhood, he knows she is heading for grannies house
  wolfEncounter : $hasnVisited * wolfNotEncountered * at wolf woods * $at redhood woods -o visit wolf granny * at wolf granny_house.

  % Any character can meet any character if they are in the same place
  meet : $hasnVisited * $at C woods * $at C' woods -o meet C C'.

  % From any location, redhood can visit granny
  visitGranny : $hasnVisited * at redhood L -o at redhood granny_house * visit redhood granny * visitToken.

  % Wolf can eat anyone that is in the same location
  eat : $hasnVisited * $at wolf L * at C L -o dead C.

  % Anyone that is in the woods has the possibility to fight the wolf
  fightWolf : $hasnVisited * at C L * at wolf L -o fightToken C L.

  % Possibility to get lost in the woods
  gets_lost : $hasnVisited * at C woods -o.

  % Possibility to wander around in the woods
  wander_in_woods : $hasnVisited * at C woods -o at C woods.
}
stage fightStage = {
    wolf_wins : fightToken C L -o at wolf L * dead C.
    character_wins : fightToken C L -o at C L * dead wolf.
}

stage visitStage = {
    % If someone visits granny, she can offer cookies
    offerCookies : $visit C granny * cookieToken -o offerCookies granny C.

    % If cookies were offered, they can be eaten
    eatCookies : offerCookies C C' -o.

    % Visit grandma when wolf doesn't know about whereabouts bc he hasn't encountered redhood
    grandma_no_wolf : visit redhood granny * wolfNotEncountered -o livedHappilyEverAfter.

    % Consume the visit token and scream
    scream : visit redhood granny * at wolf granny_house -o scream.

    % Consume the visit token and redhoods location, run away to the woods
    runAway : visit redhood granny * at wolf granny_house * at redhood L -o at redhood woods.

    % Woodcutter heard scream
    woodcutterAlerted : scream * at woodcutter woods -o at woodcutter granny_house * fightToken woodcutter granny_house.

    % Woodcutter didn't hear scream, because he was in the cabin
    woodcutterNotalerted: scream * at woodcutter cabin -o dead redhood.
}

% Stage transferring rules
qui * stage spawnStage -o stage mainStage.

qui * stage mainStage * $fightToken C L -o stage fightStage.
qui * stage fightStage -o stage mainStage.

qui * stage mainStage * visitToken -o stage visitStage.
qui * stage visitStage * $fightToken C L -o stage fightStage.

% Contexts
context init =
{live redhood woods, live mom woods, live granny granny_house, at woodcutter spawn, at wolf spawn, hasntVisited, wolfNotEncountered, smellToken, smellToken, howlToken, howlToken, cutTreeToken, cutTreeToken, cookieToken, cookieToken, turnipToken}.

#trace _ spawnStage init.
C - Simple language interpreter module example

Before execution of the simple language interpreter (common sense story 0):

--- STEP:  spawn_cabin woodcutter
  REMOVED: {(at woodcutter spawn)}
  ADDED:  {(at woodcutter cabin)}
--- STEP:  spawn_cabin wolf
  REMOVED: {(at wolf spawn)}
  ADDED:  {(at wolf cabin)}
--- STEP:  `anon0
  REMOVED: {qui, (stage spawnStage)}
  ADDED:  {(stage mainStage)}
--- STEP:  eat cabin woodcutter
  REMOVED: {(at wolf cabin), (at woodcutter cabin), hasntVisited}
  ADDED:  {(dead woodcutter), (at wolf cabin), hasntVisited}
--- STEP:  leaveHouseToWoods redhood
  REMOVED: {(live redhood woods)}
  ADDED:  {(at redhood woods)}
--- STEP:  visitGranny woods
  REMOVED: {(at redhood woods), hasntVisited}
  ADDED:  {visitToken, (visit redhood granny), (at redhood granny_house)}
--- STEP:  leaveHouseToWoods mom
  REMOVED: {(live mom woods)}
  ADDED:  {(at mom woods)}
--- STEP:  offerCookies redhood
  REMOVED: {(visit redhood granny), cookieToken}
  ADDED:  {(offerCookies granny redhood), (visit redhood granny)}
--- STEP:  eatCookies granny redhood
  REMOVED: {(offerCookies granny redhood)}
  ADDED:  {}
--- STEP:  offerCookies redhood
  REMOVED: {(visit redhood granny), cookieToken}
  ADDED:  {(offerCookies granny redhood), (visit redhood granny)}
--- STEP:  eatCookies granny redhood
  REMOVED: {(offerCookies granny redhood)}
  ADDED:  {}
--- STEP:  `anon3
  REMOVED: {qui, visitToken, (stage mainStage)}
  ADDED:  {(stage visitStage)}
--- STEP:  grandma_no_wolf
  REMOVED: {(visit redhood granny), wolfNotEncountered}
  ADDED:  {livedHappilyEverAfter}
Before execution of the simple language interpreter:
The woodcutter is in the cabin.
The wolf is in the cabin.
The wolf eats the woodcutter in the cabin.
The girl leaves the house and goes to the woods.
The girl leaves the woods to visit granny.
The mom leaves the house and goes to the woods.
Granny offers cookies to the girl.
The girl eats cookies.
Granny offers cookies to the girl.
The girl eats cookies.
They lived happily ever after.
### D - Stories presented in Study: Story generation, common sense and language interpretation

Table 2: Generated stories

<table>
<thead>
<tr>
<th>The wolf is in the cabin. The woodcutter is in the cabin. The mom leaves the house and goes to the woods. The mom gets lost in the woods. The wolf eats the woodcutter in the cabin. The girl leaves the house and goes to the woods. The girl leaves the woods to visit granny. They lived happily ever after.</th>
<th>The woodcutter is in the woods. The wolf is in the woods. The woodcutter gets lost in the woods. The mom leaves the house and goes to the woods. The mom fights the wolf in the woods. The girl leaves the house and goes to the woods. The girl leaves the woods to visit granny. The mom wins the fight against the wolf in the woods. They lived happily ever after.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The wolf is in the woods. The woodcutter is in the woods. The wolf howls at the woodcutter. The wolf howls at the woodcutter. The wolf meets the woodcutter. The woodcutter gets lost in the woods. The wolf gets lost in the woods. The mom leaves the house and goes to the woods. The mom wanders in the woods. The mom gets lost in the woods. Granny cooks turnips. The girl leaves the house and goes to the woods. The girl leaves the woods to visit granny. Granny offers cookies to the girl. The girl eats cookies. Granny offers cookies to the girl. The girl eats cookies. They lived happily ever after.</td>
<td>The wolf is in the cabin. The woodcutter is in the woods. The woodcutter wanders in the woods. The girl leaves the house and goes to the woods. The woodcutter cuts a tree. The woodcutter meets the girl. The girl wanders in the woods. The woodcutter gets lost in the woods. Granny cooks turnips. The mom leaves the house and goes to the woods. The girl wanders in the woods. The girl leaves the woods to visit granny. Granny offers cookies to the girl. The girl eats cookies. Granny offers cookies to the girl. The girl eats cookies. They lived happily ever after.</td>
</tr>
<tr>
<td>The wolf is in the woods. The woodcutter is in the cabin. The mom leaves the house and goes to the woods. The wolf eats the mom in the woods. The girl leaves the house and goes to the woods. The girl meets the wolf. The wolf howls at the girl. The wolf howls at the girl. The wolf eats the girl in the woods. The wolf gets lost in the woods.</td>
<td>The wolf is in the cabin. The woodcutter is in the cabin. The mom leaves the house and goes to the woods. The mom wanders in the woods. The wolf eats the woodcutter in the cabin. The girl leaves the house and goes to the woods. The mom wanders in the woods. The girl meets the mom. The mom gets lost in the woods. The girl gets lost in the woods. The girl loses in the woods.</td>
</tr>
</tbody>
</table>
The wolf is in the woods.
The woodcutter is in the cabin.
The girl leaves the house and goes to the woods.
The wolf meets the girl.
The wolf howls at the girl.
The girl gets lost in the woods.
The wolf wanders in the woods.
The mom leaves the house and goes to the woods.
The wolf wanders in the woods.
The mom fights the wolf in the woods.
Granny cooks turnips.
The wolf wins the fight against the mom in the woods.
The wolf wanders in the woods.
The wolf wanders in the woods.
The girl leaves the house and goes to the woods.
The girl leaves the woods to visit granny.
Granny offers cookies to the girl.
The girl eats cookies.
Granny offers cookies to the girl.
The girl eats cookies.
They lived happily ever after.

The woodcutter is in the woods.
The wolf is in the cabin.
The wolf smells the woodcutter in the cabin.
The mom leaves the house and goes to the woods.
The wolf smells the woodcutter in the cabin.
The girl leaves the house and goes to the woods.
The girl wanders in the woods.
The girl leaves the woods to visit granny.
Granny offers cookies to the girl.
The girl eats cookies.

The woodcutter is in the cabin.
The wolf is in the woods.
The girl leaves the house and goes to the woods.
The wolf meets the girl in the woods.
The girl wanders in the woods.
The mom leaves the house and goes to the woods.
The girl meets the mom.
The girl leaves the woods to visit granny.
The girl runs away from the granny house.
The woodcutter wins the fight against the wolf in the granny house.
The woodcutter hears the scream through the forest.
Table 3: Generated stories (adapted)

| Romeo and Juliet are at the chapel. Juliet strikes a bad impression on Romeo. Mercutio and Tybalt decide to go to the party. They start drinking a lot and both of them get drunk, they start fighting. They reconsider and stop fighting. Mercutio introduces one of his friends, Romeo to Tybalt. Romeo and Tybalt become friends too. Romeo gets in a fight, his new friend Tybalt defends him. | Romeo, Mercutio, Tybalt and Juliet are in town. Mercutio decides to go to the party and starts drinking. Tybalt, Juliet and Romeo join the party, both guys start drinking. Juliet leaves a bad impression on Romeo. Everyone starts drinking, Mercutio, Juliet and Tybalt get drunk. Mercutio attacks Tybalt in a drunken mood. He reconsiders and makes up with Tybalt. Romeo starts drinking heavily and gets drunk. He attacks Juliet and kills her. |
| Juliet and Tybalt decide to go to the party, Tybalt starts drinking. Mercutio decides to go to the party as well. Juliet starts drinking heavily. Tybalt gets sad because he sees Mercutio at the party, who dislikes him. Juliet continues drinking and gets drunk. Mercutio and Tybalt start drinking, Tybalt gets drunk. Tybalt gets even more sad when seeing Mercutio, he gets depressed at the party. Mercutio starts drinking a lot, he gets drunk. Mercutio and Tybalt, both drunk, start fighting at the party. Tybalt becomes suicidal and he gets killed by Mercutio in the fight. | Romeo, Mercutio, Tybalt and Juliet are in town. Romeo and Mercutio decide to go to the party, they start drinking. Juliet decides to join them at the party. She leaves a bad impression on Romeo. Tybalt joins the party as well, he starts drinking heavily. Mercutio gets drunk. Romeo and Juliet start drinking heavily as well, he attacks her. Romeo kills Juliet in their fight. Tybalt attacks Romeo, Mercutio dutifully jumps in between. He gets killed by Tybalt in this act of selflessness. |
| Mercutio decides to go to the party. Romeo is at home with Juliet, she leaves a bad impression on him. She gets sad because Romeo dislikes her. Mercutio starts drinking heavily. Juliet leaves home for the party. Mercutio continues drinking and gets drunk. Romeo leaves home for the party. Juliet gets sad seeing him there, she starts drinking heavily. Romeo starts drinking as well. | Mercutio decides to go to the party. Romeo is at home with Juliet, she leaves a bad impression on him. She gets sad because Romeo dislikes her. Mercutio starts drinking heavily. Juliet leaves home for the party. Mercutio continues drinking and gets drunk. Romeo leaves home for the party. Juliet gets sad seeing him there, she starts drinking heavily. Romeo starts drinking as well. |
Juliet gets even more sad, she becomes depressed. Romeo falls in love with drunk Mercutio. Juliet continues drinking heavily and gets drunk. She becomes suicidal at the party and kills herself. Romeo and Mercutio are in love, they leave the party for the chapel and get married.

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Generated stories (general)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo and Juliet decide to go to the party. Romeo starts liking Juliet, he falls in love with her. He takes her to the chapel in a loving mood and marries her there. They head back to the party, where he falls in love with her again.</td>
<td>Juliet and Tybalt decide to go to the party, Tybalt starts drinking. Mercutio decides to go to the party as well. Juliet starts drinking heavily. Tybalt gets sad because he sees Mercutio at the party, who dislikes him. Juliet continues drinking and gets drunk. Mercutio and Tybalt start drinking, Tybalt gets drunk. Tybalt gets even more sad when seeing Mercutio, he gets depressed at the party. Mercutio starts drinking a lot, he gets drunk. Mercutio and Tybalt, both drunk, start fighting at the party. Tybalt becomes suicidal and he gets killed by Mercutio in the fight.</td>
</tr>
</tbody>
</table>