MARKUP LANGUAGE FOR PSYCHOLOGICAL ARCHETYPES

Alexandru BUZU[1*], Bogdan ZLATOVCEN[1], Dumitru MORARU[1], Maria LEȘENCO[1], Marius BOTEZATU[1]

[1]*Faculty of Computers, Informatics and Microelectronics, Software Engineering, Group FAF-212, Technical University of Moldova, Chișinău, The Republic of Moldova*

*Corresponding author: Alexandru Buzu, alexandru.buzu@isa.utm.md

Coordinator: Vasili BRAGA, conf. univ, TUM

Abstract. *The main topic of this scientific article is the basics for development of a markup language based on NLP. The proposed solution is oriented to help content creators, and especially game developers, to create original behavior and background history for NPCs and other characters. The markup language aims to have a very user friendly syntax, and operate with ChatGPT API to give original and AI based responses.*

Keywords: *formal language, grammar, AI, game development, NPC.*

Introduction

An AI-based markup language for NPC stories can revolutionize the way game developers create immersive and engaging game experiences. By automating the process of NPC story generation, an AI-based markup language can significantly reduce the amount of time and resources required to create unique and high-quality stories. This can be especially advantageous for indie game developers and smaller studios with limited resources, enabling them to focus on other critical aspects of game development.

Moreover, AI-powered language models can generate dynamic NPC stories that respond dynamically to player actions, resulting in more engaging and immersive game experiences. The variability of the generated stories can lead to increased replayability for players, as each playthrough may have different NPC storylines. This can significantly improve the quality and variability of the game, making it more enjoyable for players.

Additionally, an AI-based markup language for NPC stories can provide game developers with a platform for experimentation and innovation in storytelling. With the ability to generate dynamic NPC stories based on player input, developers can explore new narrative structures and gameplay mechanics, leading to the creation of more innovative and engaging games. This can open up new opportunities for developers to experiment with different narrative styles and explore new and creative storytelling techniques.

Although there are several existing tools and technologies that game developers use to create NPC stories, including markup languages like Ink and Yarn, an AI-based markup language that generates NPC stories automatically is a relatively new concept. The use of AI and machine learning in game development is becoming increasingly common, as developers seek to create more immersive and engaging game experiences for players.

An example of an existing solution is AI Dungeon - a text-based adventure game that uses natural language processing and machine learning to create dynamic storylines based on user input. It enables users to create custom characters and storylines, which are generated using an AI-powered language model [1]. Artie, on the other hand, is an AI-based storytelling platform that enables users to create interactive stories with animated characters. Artie's AI engine generates stories based on user input, creating custom narratives for each character based on their traits and personalities.

Overall, an AI-based markup language for NPC stories should provide game developers with a powerful tool to create immersive and engaging game experiences for players. It should enable the creation of dynamic NPCs that respond to player choices and actions, while also being accessible to developers with varying levels of experience in programming and game development. The AI model used by the markup language should be well-trained and maintained, with regular updates and improvements to ensure high-quality output. By leveraging the power of AI and machine learning, game developers can create truly immersive and engaging game experiences that captivate and entertain players for hours on end.

Language overview

- Basic Computation: The basic computation that this DSL performs is data manipulation and analysis related to psychological archetypes.
- Data Structures: The basic data structures in this DSL are archetypes and their attributes, which can be created and manipulated by the user.
- Control Structures: The user can specify control flow using different key parameters and their attributes.
- Input and Output: A program in this DSL requires input in the form of archetypal data and produces output in the form of analysis, manipulation of that data and a virtual chat with the described person.
- Error Handling: Programs may go wrong due to invalid input or incorrect use of syntax. The language will communicate these errors to the user through informative error messages.
- Other DSLs for this Domain: There are currently no widely-used DSLs specifically for the domain of psychological archetypes. However, there are general-purpose programming languages that can be used to manipulate archetypal data, such as Python or R. This language will be more user-friendly and focused on the needs of archetypal data analysis.

Grammar

For a markup language for psychological archetypes, we can define its grammar as follows:
L(G) = (S, P, VN, VT), where:

- S is the start symbol
- P is a finite set of production rules
- VN is a finite set of non-terminal symbols
- VT is a finite set of terminal symbols

The terminal symbols represent the actual content of the document or data being marked up. The non-terminal symbols represent the structure or syntax of the markup language itself.
The production rules describe how non-terminal symbols can be replaced or expanded into other non-terminal or terminal symbols [2]. These rules define the syntax and structure of the markup language.
To specify the grammar representation for the markup language we need to use meta notations:

Table 1

Meta notation

Symbol	Definition
< >	Non-Terminal symbol
*	Zero or more occurrences
+	One or more occurrences
|	Separates the alternative symbols
→	Derivation
~	Except: any character except

The grammar representation for markup language project:
<markup-language> → <description> <setting> <response>
<description> → **Description** "{" <description-property>* "}"
<description-property> → <name-property> | <type-property> | <mbti-property> |<role-property> | <background-property>
<name-property> → **name** = <string-literal>
<type-property> → **type** = <string-literal>
<mbti-property> → **mbti =** <string-literal>
<role-property> → **role =** <string-literal>
<background-property> → "background" "=" <string-literal>
<setting> → **Setting** "{" <setting-property>* "}"
<setting-property> → <type-property> | <category-property> | <background-property>
<category-property> → **category =** "[" <string-literal> ("," <string-literal>) * "]"
<response> → **Response** "{" <response-property>* "}"
<response-property> → <length-property> | <prompt-property>
<length-property> → **length =** <number-literal>
<prompt-property> → **prompt =** <string-literal>
<number-literal> → <digit>+
<digit> → 0 | 1 | 2 | ... | 9
<string-literal> → <char>*
<char> → <digit> | a | b | c | … | z | A | B | C | … | Z | - | _ | , | ; | " | " | : | . | ' | ? | !

An example of code using this grammar representation is:

```
Description {
        name=John
        type=NPC
        mbti=intj
        role=protagonist
        background="John is a witch hunter"
}
Setting {
        type=game
        category=[adventure, magic, open-world]
        background="A world similar to Mars, where witches are living in forests and fight with
humans"
}
Response {
        length=300
        prompt="I need the background story of the character. Add some info about his family. Give
3 possible stories in which John is a side character"
}
```

Conclusions

In a world where automation takes a vital role in many fields, it is important for content creators to have various tools which will simplify some routine processes and boost their creativity and performance. The niche of DSL and Markup languages for defining NPC behavior and other topics related to creating the world, is not occupied at all, and the content creators are in need of solutions. That is why, the development of a markup language which will define the NPCs and other characters, is a very practical solution, and not so hard to realize with the help of tools like ChatGPT API.

References

1. AI Dungeon. [online]. [accessed 10.03.2023]. Available: https://aidungeon.io
2. TOMASSETTI, G. *A Guide to Parsing: Algorithms and Terminology* [online]. [accessed 10.11.2023]. Available: https://tomassetti.me/guide-parsing-algorithms-terminology/