DSL FOR AI PROJECTS ARCHITECTURE

Lucian LUPAN, Mihail MIHALACHI, Nicolai NEJINȚEV*,
Maria CUCOȘ, Valeria FRUNZA

Department of Software Engineering and Automatics, Group FAF-221, Faculty of Computers, Informatics and Microelectronics, Technical University of Moldova, Chisinau, Republic of Moldova

*Corresponding author: Nicolai Nejințev, nicolai.nejintev@isa.utm.md

Scientific coordinator: **Vasili BRAGA**, university assistant

Abstract. *This paper addresses the complexity of Artificial Intelligent (AI) system design and deployment by advocating for a Domain-Specific Language (DSL) specifically designed for AI projects architecture. This research clarifies the possible advantages and difficulties of implementing a DSL for AI projects through creating an opportunity for organizations to stimulate innovation, shorten development cycles, and take advantage of the growing opportunities in the AI landscape by adopting a DSL designed specifically for AI.*

Keywords: *Machine Learning, architecture, Domain-Specific Language, AI, Speech-to-Text, Optical Character Recognition*

Introduction

In the swiftly evolving landscape of AI, navigating the intricacies of designing and deploying AI systems presents formidable challenges. As development of numerous products incorporating Machine Learning (ML) and other AI components continues to progress, various challenges have been encountered, particularly concerning communication and alignment among team members. These teams have typically been multidisciplinary, encompassing profiles beyond traditional developers and software engineers, including data scientists, psychologists, and AI experts [1].

This paper advocates for the adoption of SOLeAI, a Domain-Specific Language (DSL) meticulously crafted for AI project architecture. SOLeAI leverages cutting-edge technologies such as Speech-to-Text (STT) and Optical Character Recognition (OCR), enabling seamless integration of spoken and written language processing capabilities into AI projects. By using SOLeAI, organizations can enjoy many benefits like increased productivity, smoother teamwork, automated processes, and faster development [1].

Grammar

In the realm of programming languages, understanding the interplay between semantics and syntax is paramount. A programming language is described by the combination of its semantics and its syntax. The semantics provide the meaning of every construction that is possible in the chosen programming language [2]. Complementing semantics, syntax provides the structural rules governing the arrangement and composition of language elements.

A grammar transforms the program, which is normally represented as a linear sequence of ASCII characters, into a syntax tree [2]. Central to the comprehension and manipulation of programming languages is the notion of grammar. By encapsulating the syntactic rules and constraints of a programming language, grammar plays a pivotal role in enabling software developers to express their ideas effectively and concisely. The presented DSL's grammar notations are precisely outlined in Tab. 1.

Table 1

Grammar notations

Notation	Description
⟨foo⟩	foo is a non-terminal symbol
foo	**foo** is a terminal symbol
[x]	zero or more occurrences, x is an optional parameter, '[' and ']' are terminal symbols.
x*	zero or more occurrences of x
x^+	one or more occurrences of x
{ }	a grouping, '{' and '}' in quotes are terminal symbols
|	alternative option

$G = (S, V_n, V_t, P)$
S - start symbol
V_n - finite set of non-terminal symbols
V_t - finite set of terminal symbols
P - finite production rules

S = {⟨prog⟩}

V_n = {⟨prog⟩, ⟨sorb⟩, ⟨block⟩, ⟨ifBlock⟩, ⟨statement⟩, ⟨pipeStream⟩, ⟨varAssignment⟩, ⟨varName⟩, ⟨value⟩, ⟨function⟩, ⟨functionName⟩, ⟨inputData⟩, ⟨comment⟩, ALPHANUM}

V_t = {NEWLINE, {, }, if, (,), =, ->, //, STRING, COMMENT_STRING}

P = { <prog> -> ⟨sorb⟩* EOF

<sorb> -> NEWLINE* ⟨statement⟩* $NEWLINE^+$ | NEWLINE* ⟨block⟩ NEWLINE*
<block> -> '{' (sorb | NEWLINE)* '}'
<ifBlock> -> 'if' '(' ⟨value | pipeStream⟩')' ⟨statement | block⟩
<statement> -> ⟨varAssignment | ifBlock | pipeStream | comment⟩
<pipeStream> -> ⟨function | inputData⟩('->' ⟨function⟩)*
<varAssignment> -> ⟨varName⟩ '=' ⟨value|pipeStream⟩
<varName> -> ALPHANUM
<value> -> STRING | ⟨varName⟩
<function> -> ⟨functionName⟩ ⟨value⟩*
<functionName> -> ALPHANUM
<inputData> -> 'messageText' | 'messageImage' | 'messageAudio'
<comment> -> COMMENT_STRING

STRING -> '"' ~["\r\n]* '"'
COMMENT_STRING -> '//' ~[\r\n]*
ALPHANUM -> [a-zA-Z]+[a-zA-Z0-9]*
NEWLINE -> ('r'?'\n')+
WHITESPACE -> (' ' | '\t')+ -> skip

Assignment

In SOLeAI, assignments follow a straightforward process. As long as the user provides valid data input, the program interprets that value and assigns it to the respective keyword, following the structure of the grammar. The only user-defined assignment function allowed in the grammar is <varAssignment>, a function which enables the user to create a variable, storing either a <value> a <pipeStream> in memory, granting the ability to reuse it later. The restriction for assigning is that variables must be declared and assigned before they're used. This ensures that variables are properly initialized before being utilized within the program.

Used technologies

STT technology, also referred to as automatic speech recognition (ASR), is a transformative tool converting spoken language into written text, finding applications across diverse industries. Acoustic Analysis initiates the process by capturing audio input via microphones, analyzing the audio signal to extract features like frequency, amplitude, and duration [3]. Language models refine STT output by considering word sequence probabilities in natural language, aiding in disambiguation and improving accuracy. Post-processing techniques like grammar checking and error correction further enhance transcription accuracy.

STT technology enhances accessibility for individuals with disabilities, facilitates transcription services for audio recordings, enables hands-free operation of devices via virtual assistants, and powers interactive voice response systems and call center automation. It also supports real-time translation services and voice-enabled search engines. Whisper, an ASR system that will be used in further implementation, stands out with its robustness to accents, background noise, and technical language, enabled by training on a large and diverse dataset, facilitating transcription in multiple languages and translation into English [4].

OCR is the process that converts an image of text into a machine-readable text format [5]. This technology serves as a transformative solution for converting various document formats, including scanned paper documents, PDF files, and digital images, into editable and searchable data. OCR systems undertake a series of preprocessing steps, including noise reduction, binarization, and deskewing, to enhance the quality and accuracy of the document image. Following preprocessing, OCR systems employ pattern recognition and ML algorithms to locate and segment text regions within the document image. One of the significant advantages of OCR technology is its ability to digitize paper documents, such as books, newspapers, and archival materials, thereby facilitating electronic storage and accessibility [5].

PyTesseract, a Python wrapper for Google's Tesseract-OCR Engine, stands out as a prominent tool to be used for integrating OCR capabilities into Python applications. With Tesseract's robust open-source OCR engine at its core, PyTesseract offers a seamless and versatile solution for incorporating OCR functionality into diverse projects and workflows.

Other DSLs

Several process modeling languages exist, including Business Process Model and Notation (BPMN) [6] and Software & Systems Process Engineering Metamodel (SPEM) [7], along with their extensions. While SPEM serves as an Object Management Group (OMG) [8] standard for delineating software development processes, it deliberately lacks distinct features tailored to specific domains or disciplines, such as AI. To the authors' knowledge, none of the existing process modeling languages offer AI-specific extensions. In the realm of DSLs for AI, various languages cater to modeling specific AI activities, such as OptiML [9], Arbiter [10], or Pig Latin [11]. However, none of these DSLs prioritize process-related aspects.

Conclusions

In conclusion, the development and deployment of AI systems present multifaceted challenges in today's dynamic technological landscape. Throughout this paper, it has been explored the significance of adopting Domain-Specific Languages tailored specifically for AI projects architecture. By delving into the complexities of SOLeAI, the document has elucidated the potential advantages it offers in addressing the complexities of AI system design and deployment. Through the investigation, the paper highlights the potential of SOLeAI to enhance productivity, foster collaboration among multidisciplinary teams, and expedite development cycles in AI projects.

As organizations endeavor to fully leverage the capabilities of AI technologies, the adoption of specialized DSLs such as SOLeAI stands out as a strategic necessity. Through the integration of OCR and STT functionalities, SOLeAI empowers organizations to streamline the development of AI systems tailored to image and audio data processing. Looking ahead, continued research and experimentation will be essential to refine and expand the capabilities of SOLeAI, driving transformative advancements in AI system architecture and deployment.

References

[1] S. Morales, R. Claris, and J. Cabot, *Towards a DSL for AI Engineering Process Modeling,* in International Conference on Product-Focused Software Process Improvement (PROFES 2022).

[2] *Introduction to Programming Languages/Grammars,* September 15 2016 [Online] [Accessed: 09.04.2024]. Available: https://en.wikibooks.org/wiki/Introduction_to_Programming_Languages/Grammars

[3] *Audio Analysis With Machine Learning: Building AI-Fueled Sound Detection App,* May 12 2022 [Online] [Accessed: 12.04.2024]. Available: https://www.altexsoft.com/blog/audio-analysis/

[4] *Introducing Whisper* [Online] [Accessed: 12.04.2024]. Available: https://openai.com/research/whisper

[5] *What is OCR (Optical Character Recognition)?* [Online] [Accessed: 12.04.2024]. Available: https://aws.amazon.com/what-is/ocr/

[6] *Object Management Group Business Process Model and Notation,* 1997 [Online] [Accessed: 12.04.2024]. Available: https://www.bpmn.org/

[7] *About the software & systems process engineering metamodel specification version 2.0* [Online] [Accessed: 12.04.2024]. Available: https://www.omg.org/spec/SPEM/2.0/About-SPEM

[8] *Mission & Vision* [Online] [Accessed: 12.04.2024]. Available: https://www.omg.org/about/

[9] A.K. Sujeeth, H. Lee, K.J. Brown, T. Rompf, H. Chafi, M. Wu, A.R. Atreya, M. Odersky, and K. Olukotun, *OptiML: An implicitly parallel domain-specific language for machine learning.*

[10] J. Zucker, M. d'Leeuwen, *Arbiter: A Domain-Specific Language for Ethical Machine Learning,* February 07 2020

[11] C. Olston, B. Reed, U. Srivastava, R. Kumar, A. Tomkins, *Pig Latin: A not-so-foreign language for data processing,* June 09 2008