A generic LSTM neural network architecture to infer heterogeneous model transformations*

Lola Burgueño[1][0000−0002−7779−8810], Jordi Cabot[2][0000−0003−2418−2489], Shuai Li[3], and Sébastien Gérard[3][0000−0003−0295−0520]

[1] Universitat Oberta de Catalunya, España {lburguenoc,rclariso}@uoc.edu
[2] ICREA, España jordi.cabot@icrea.cat
[3] CEA LIST, Francia {Shuai.LI,Sebastien.GERARD}@cea.fr

Keywords: Model manipulation, Code generation, Model transformation, Artificial intelligence, Machine learning, Neural networks
Published in: Software and Systems Modeling, Vol. 21, No. 1, pp.139–156, 2022
Impact Factor: JCR 2.211 - Q3 - Position: 57/110 - Area: Computer Science / Software Engineering
DOI: https://doi.org/10.1007/s10270-021-00893-y

Abstract. Models capture relevant properties of systems. During the models' life-cycle, they are subjected to manipulations with different goals such as managing software evolution, performing analysis, increasing developers' productivity, and reducing human errors. Typically, these manipulation operations are implemented as model transformations. Examples of these transformations are (i) model-to-model transformations for model evolution, model refactoring, model merging, model migration, model refinement, etc., (ii) model-to-text transformations for code generation and (iii) text-to-model ones for reverse engineering. These operations are usually manually implemented, using general-purpose languages such as Java, or domain-specific languages (DSLs) such as ATL or Acceleo. Even when using such DSLs, transformations are still time-consuming and error-prone. We propose using the advances in artificial intelligence techniques to learn these manipulation operations on models and automate the process, freeing the developer from building specific pieces of code. In particular, our proposal is a generic neural network architecture suitable for heterogeneous model transformations. Our architecture comprises an encoder–decoder long short-term memory with an attention mechanism. It is fed with pairs of input–output examples and, once trained, given an input, automatically produces the expected output. We present the architecture and illustrate the feasibility and potential of our approach through its application in two main operations on models: model-to-model transformations and code generation. The results confirm that neural networks are able to faithfully learn how to perform these tasks as long as enough data are provided and no contradictory examples are given.

* Supported by the Spanish Government (FEDER/Ministerio de Ciencia e Innovación–Agencia Estatal de Investigación) under projects PID2021-125527NB-I00 and TED2021-130523B-I00 and Universidad de Málaga.