Anne Koziolek, Ina Schaefer, Christoph Seidl (Hrsg.): Software Engineering 2021,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2021 117

Programming in Natural Language with *fu*SE: Synthesizing Methods from Spoken Utterances Using Deep Natural Language Understanding

Sebastian Weigelt[1] Vanessa Steurer[2] Tobias Hey[1] Walter F. Tichy[1]

Abstract: With *fu*SE laypeople can create simple programs: one can teach intelligent systems new functions using plain English. *fu*SE uses deep learning to synthesize source code: it creates method signatures (for newly learned functions) and generates API calls (to form the body). In an evaluation on an unseen dataset *fu*SE synthesized 84.6% of the signatures and 66.9% of the API calls correctly[3].

Keywords: Programming in Natural Language; End-User Programming; Deep Learning; AI; NLP

Introduction: Intelligent systems became rather smart lately. One easily arranges appointments by talking to a virtual assistant or controls a smart home through a conversational interface. For the time being, users can only access built-in functionality. However, they will soon expect to add new functionality themselves. For humans, the most natural way to communicate is by natural language. Thus, future intelligent systems must be programmable in everyday language. We propose to apply deep natural language understanding to the task of synthesizing methods from spoken utterances. *fu*SE combines deep learning techniques with information retrieval and knowledge-based methods to grasp the user's intent.

Approach: *fu*SE is a system for programming in (spoken) natural language: laypersons can create method definitions by using natural language only. To investigate how laypersons teach new functionality we ran a preliminary study in which subjects were supposed to teach new skills to a humanoid robot. The study consists of four scenarios in which a humanoid robot should be taught a new skill: greeting someone, preparing coffee, serving drinks, and setting a table for two. We used the online micro-tasking platform *Prolific*[4] and were able to gather 3168 descriptions from 870 participants. Based on the findings of the preliminary study we develop the following three-tiered approach (see Figure 1). First, *fu*SE classifies teaching efforts, i.e. it determines whether an utterance comprises an explicitly stated teaching intent or not. Second, it classifies the semantic structure, i.e. *fu*SE analyzes (and labels) the semantic parts of a teaching sequence. Teaching sequences are composed of a *declarative* and a *specifying* part as well as superfluous information. Third, *fu*SE synthesizes methods, i.e. it builds a model that represents the structure of methods from syntactic

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany, {weigelt|hey|tichy}@kit.edu
[2] inovex GmbH, Karlsruhe, Germany, vsteurer@inovex.de
[3] This contribution is a short version of the paper originally published in the proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (2020) [We20].
[4] Prolific: `https://www.prolific.co/`

doi:10.18420/SE2021_46

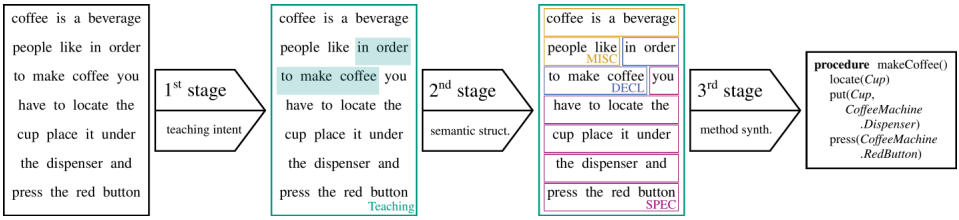

Fig. 1: Schematic overview of *fu*SE's three-tiered approach.

information and classification results. Then, it maps the actions of the specifying part to API calls, injects control structures to form the body and creates the method signature. The first two stages are classification problems. For the first we compared classical machine learning techniques, such as logistic regression and support vector machines, with neural network approaches including the pre-trained language model *BERT* [De19]. For the second task we narrow down to neural networks and *BERT*. However, for both tasks *BERT*-based models performed best: test set accuracy 97.7% (1st stage) resp. 97.3% (2nd stage). The third stage is a combination of syntactic analysis, knowledge-based techniques and information retrieval. We use semantic role labeling, coreference analysis, and a context model to build a semantic model. Afterwards, we synthesize method signatures heuristically and map instructions from the body to API calls using ontology search methods and datatype analysis. To cope with spontaneous (spoken) language, our approach relies on shallow NLP techniques only.

Evaluation: To measure the performance of *fu*SE on unseen data, we set up a case study. We created two new scenarios and used *Prolific* to collect 202 descriptions, of which we randomly drew 100; 78 of these comprise a teaching intent. In sum, the descriptions require the generation of 473 API calls. *fu*SE synthesized 73 method signatures; five were missed due to an incorrect first-stage classification. Out of 73 signatures we assessed only seven to be inappropriate. The generation of API calls (that form the method bodies) also performs well (F_1: 66.9%). These results are promising; however, we plan to improve *fu*SE with a dialog module to query the user in case of ambiguities.

Bibliography

[De19] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp. 4171–4186, June 2019.

[We20] Weigelt, Sebastian; Steurer, Vanessa; Hey, Tobias; Tichy, Walter F.: Programming in Natural Language with fuSE: Synthesizing Methods from Spoken Utterances Using Deep Natural Language Understanding. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Online, pp. 4280–4295, July 2020.