The Natural Language Programming (NLPRO) Project: Turning Text into Executable Code

Reut Tsarfaty
The ONLP Research Lab @ The Open University of Israel
reutts@openu.ac.il

Abstract

In this paper we present the *natural language programming* (NLPRO) project (via ERC-StG-2015 grant 677352), where we strive to automatically translate requirements documents directly into the executable code of the systems they describe. To achieve this, we embrace the ambiguity of NL requirements and define a three-fold research agenda wherein we (i) formalize text-to-code translation as a structure prediction task, (ii) propose a formal semantic representation in terms of Live Sequence Charts (LSCs), and (iii) develop and comparatively evaluate novel sentence-based vs. discourse-based models for semantic parsing of requirements documents, and test their accuracy on various case studies. The empirical results of our first research cycle show that the discourse-based models consistently outperform the sentence-based models in constructing a system that reflects the requirements in the document. We conjecture that the formal representation of LSCs, the joint sentence-discourse modeling strategy, and the statistical learning component, are key ingredients for effectively tackling the NLPRO long-standing challenge.

1 NLPRO Project Statement: A Tale of Two Disciplines

Can we program computers in our native tongue? This idea, termed *natural language programming*, has attracted attention almost since the inception of computers themselves [Dij79]. From the point of view of software engineering (SE), efforts to program in natural language (NL) have relied thus far on controlled natural languages (CNL) — small unambiguous fragments of English with strict grammars and limited expressivity. *Is it possible to replace CNLs with truly natural, human language?* From the point of view of natural language processing (NLP), current technology successfully extracts unambiguous facts from ambiguous NL texts. However, human-like NL understanding goes far beyond fact extraction — it requires dynamic interpretation processes which affect, and are affected by, the environment, update states and lead to action. So, *is it possible to endow computers with this kind of NL understanding?*

These two questions are fundamental to SE and NLP, respectively, and addressing each requires a huge leap forward in the respective field. In this project we suggest that the solutions to these seemingly separate challenges are actually closely intertwined, and that one community's challenge is the other community's stepping stone for a huge leap, and vice versa. Specifically, we propose to view *executable scenarios* in SE as *semantic structures* in NLP, and use them as the basis for broad-coverage semantic parsing.

The ambitious cross-disciplinary goal of this project is to develop an NL compiler which will accept an NL description as input and return an executable system as output. Moreover, it is intended to continuously improve its NL understanding capacity via online learning that will feed on verification, simulation, synthesis or user feedback. Such NL compilers will have vast applications in AI, SE, robotics and cognitive computing, and they have the potential to fundamentally change the way humans and computers interact.

2 Background: Requirements Engineering using Controlled Languages

Requirements elicitation is a process whereby a system analyst gathers information from a stakeholder about a desired system to be implemented. The knowledge collected by the analyst may be *static*, referring to the conceptual model (the entities, properties, possible values) or *dynamic*, referring to the behavior that the system should follow (who does what to whom, when, how, etc) [NE00]. A stakeholder interested in the system typically has a specific static and dynamic domain in mind, but he or she cannot necessarily prescribe any formal models or code artifacts. The term *requirements elicitation* I use here refers to a piece of discourse in natural language, by means of which a stakeholder communicates their desiderata to the analyst. The role of the system analyst is to understand the different requirements and transform them into formal constructs, diagrams or executables. Moreover, the analyst needs to consolidate the different pieces of information to uncover a single shared domain.

To streamline this process, studies in software engineering aim to develop intuitive symbolic systems, often termed *controlled natural languages* (CNL), with which human agents can encode requirements which would then be unambiguously translated into formal or executable artifacts [FS95, BL02]. Gordon and Harel [GH09] for instance define a CNL that can be used for specifying requirements which can be effectively translated into *live sequence charts* (LSC) [DH01, HM03], a formal language for specifying the dynamic behavior of reactive systems. However, the grammar that underlies this language fragment is highly ambiguous, and all disambiguation has to be conducted manually by human intervention, which in turn makes the process slow, unintuitive, and time consuming. This particular aspect of their work reflects a rather general phenomenon: the more natural a symbolic system, or a CNL, is, the harder it is to develop an unambiguous translation engine for it [Kuh14]. As a result, much work on CNL parsing for requirements documents requires a human in the loop, or instead, drastically narrowing down the space of allowed utterances, in order to avoid any ambiguity.

3 The Proposal: Semantic Parsing using Content and Context

In this project we accept the ambiguity of requirements descriptions as a premise, and aim to directly address the challenge of automatically recovering a single unambigous formal representation of the complete system by parsing the requirements document — one that *best* reflects the human-perceived interpretation of the document.

Recent advances in natural language processing (NLP) are define semantic parsing as the task of automatically assigning informal natural language utterances with a formal, unambiguous representation of meaning. Their formal representation output type is often task-specific — for example, Zettelmoyer and Collins [ZC05], Liang et al [LJK11], Artzi and Zettlemoyer [AZ13], and Liang and Potts [LP14], use different formalisms and various kinds of statistical learning signals to support the automatic assignment of meaning in different tasks. In particular, the model of Lei et al [LLBR13] induces input parsers from format descriptions, and Kushman and Barzilay [KB13] automatically convert free textual descriptions into regular expressions. However, these models interpret only short and local instructions, and rarely do they take into account the entire document, in order to deliver a formal description of a complete system.

Here we cast the requirements documents interpretation task as a structure prediction task, where we accept a piece of discourse as input and aim to automatically predict a formal model of the static and dynamic domain as output. We currently assume that the input requirements document is given in the simple, yet highly ambiguous, fragment of English of Gordon and Harel [GH09].[1] The output, in contrast, is a sequence of unambiguous and well-formed formal constructs that represent the dynamic behavior of the system, called *live sequence charts* (LSC) [DH01, HM03] tied to a single shared code-base called a *system model* (SM).

The key idea we promote in this work is that discourse context provides substantial disambiguating information for the semantic interpretation of individual requirements in the document. We present a novel system for automatically translating the requirements into executable artefacts based on a *joint* sentence-level and discourse-level probabilistic generative model. The solution we present takes the form of a hidden markov model (HMM) where emission probabilities are calculated in CKY charts to reflect the grammaticality and interpretability of each individual textual requirements via a *probabilistic grammar*, and transition probabilities model the overlap between SM snapshots of a single, shared, domain. Using efficient viterbi decoding, we search for the best sequence of domain snapshots that has most likely generated the entire document. We empirically show that such a joint model consistently outperforms a sentence-based model learned from the same set of data.[2]

[1]This version assumes a fragment of English grammar with a closed set of function words and an unlimited lexicon of open class categories (verbs, nouns and adjectives).

[2]For further information on the model as well as thorough experimental results, we kindly refer the reader to Tsarfaty et al [TPW$^+$14] and references therein.

4 Natural Language Programming: Conclusions and Perspectives

The contribution of the first development cycle of the NLPRO project is three-fold: (i) we formalize the text-to-code translation as a structure prediction task, (ii) we propose a formal semantic representation with well-defined grounding for RE, and (iii) we empirically evaluate sentence-based and discourse-based models for semantic parsing of requirements. We show a consistent improvement of discourse-based over sentence-based models, in all case studies. In the future, we intend to extend this model for interpreting requirements in unrestricted English, endowed with a more sophisticated discourse interpretation function.

All in all, the automatic interpretation of requirements documents presents an exciting challenge for both natural language processing (NLP) and software engineering (SE). In NLP, effectively addressing text-to-code translation requires fundamentally rethinking the nature of semantic representation, the scope of interpretation, and how the automatic discovery the entities, actions, conditions, temporal markers, constraints, execution modalities, etc., can lead to a single coherent storyline/system description. In SE, the availability of tools for automatic analysis and interpretation of requirements has the potential of introducing a huge leap in the speed and accuracy of system development. It is hoped that this prospect will lead to new and novel methodologies of requirements engineering, in which man and machine offer respective and complementary contribution towards maximally effective and efficient development.

Acknowledgements

This work is kindly supported by the European Research Council, ERC-StG-2015 grant 677352, for which we are grateful.

References

[AZ13] Y. Artzi and L. S. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1:49–62, 2013.

[BL02] B. Bryant and B. S. Lee. Two-level grammar as an object-oriented requirements specification language. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS02)*. IEEE Computer Society, 2002.

[DH01] W. Damm and D. Harel. LSCs: Breathing life into message sequence charts. *Form. Methods Syst. Des.*, 19(1):45–80, July 2001.

[Dij79] Edsger W. Dijkstra. On the foolishness of ”natural language programming”. In FriedrichL. Bauer, Manfred Broy, E.W. Dijkstra, S.L. Gerhart, D. Gries, M. Griffiths, J.V. Guttag, J.J. Horning, S.S. Owicki, C. Pair, H. Partsch, P. Pepper, M. Wirsing, and H. WÃssner, editors, *Program Construction*, volume 69 of *Lecture Notes in Computer Science*, pages 51–53. Springer Berlin Heidelberg, 1979.

[FS95] N. E. Fuchs and R. Schwitter. Attempto: Controlled natural language for requirements specifications. In Markus P. J. Fromherz, Marc Kirschenbaum, and Anthony J. Kusalik, editors, *LPE*, 1995.

[GH09] M. Gordon and D. Harel. Generating executable scenarios from natural language. In *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing ’09, pages 456–467, Berlin, Heidelberg, 2009. Springer-Verlag.

[HM03] D. Harel and R. Marelly. *Come, Let’s Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[KB13] N. Kushman and R. Barzilay. Using semantic unification to generate regular expressions from natural language. In *Proceedings of NAACL-HLT*, pages 826–836, 2013.

[Kuh14] T. Kuhn. A survey and classification of controlled natural languages. *Computational Linguistics*, 40(1):121–170, 2014.

[LJK11] P. Liang, M. I. Jordan, and D. Klein. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599, 2011.

[LLBR13] T. Lei, F. Long, R. Barzilay, and M. C. Rinard. From natural language specifications to program input parsers. In *ACL (1)*, pages 1294–1303, 2013.

[LP14] P. Liang and C. Potts. Bringing machine learning and compositional semantics together. *Annual Reviews of Linguistics (submitted)*, 0, 2014.

[NE00] B. Nuseibeh and S. Easterbrook. Requirements engineering: A roadmap. In *Proceedings of ICSE*, 2000.

[TPW$^+$14] R. Tsarfaty, E. Pogrebezky, G. Weiss, Y. Natan, S. Szekely, and D. Harel. Semantic parsing using content and context: A case study from requirements elicitation. In *Proceedings of EMNLP*, pages 1296–1307, 2014.

[ZC05] L. S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666. AUAI Press, 2005.