Modelling Natural Language, Programs, and their Intersection

Graham Neubig (Carnegie Mellon University), Miltiadis Allamanis (Microsoft Research)

Description

Just like natural language is a tool that humans use to communicate with each-other, programming languages are tools that humans use to communicate with computers. Because of the increasing need for programs and programming in our working and everyday lives, there are now massive amounts of source code being produced every day. As a result, it is ever more important for an ever increasing segment of the populace to be able to understand and create programs to do what they would like to do. However, programming is a specialized skill, IT education is hard-pressed to make up for this demand.

One key insight that can help us tackle this problem is that source code is *bimodal*. While one modality is targeted towards explicitly instructing the hardware on the actions to perform, the other is targeted towards the humans that need to read, understand, maintain and extend the code. Given that it is humans that are producing the software, the human-oriented modality is very strong and often takes the form of natural language: from natural language identifiers, such as variable and method names, to code comments and natural language documentation.

As a result, there is recently a burgeoning interest in research that connects natural language with the programming language artifacts. This research area has the potential to improve the efficiency and ease of programming by making connections to natural language, which is (in general) easier for humans to understand and communicate with, particularly humans who are not yet well-versed in programming. Some examples of relevant tasks include:

- **Automatic explanation of programs in natural language (code-to-language):** Highly connected with the task of grounded natural language generation in the NLP community, this is the task of generating natural language explanations for source code artifacts, which will allow them to be understood more easily.
- **Automatic generation of programs from natural language specifications (language-to-code):** Highly connected with the task of semantic parsing in the NLP community, this is the task of translating natural language into code that allows for grounded executable representations of natural language. This also encompasses natural language code search, which retrieves relevant code snippets based on natural language queries.
- **Modelling the natural language elements of source code:** As mentioned above, much of source code itself contains elements that are expressed in natural language (e.g. variable names and code comments), giving a form of grounded semantics to these aspects of code.
- **Analysis of communication in collaborative software development communities:** The process of developing software, particularly in multi-party projects, is a collaborative act, and as a result, provides a rich source of data for analysis of grounded communication in collaborative environments, which can then be used to improve productivity in these environments.

Proceedings of NAACL-HLT 2018: Tutorial Abstracts, pages 1–3
New Orleans, Louisiana, June 1, 2018.

In this tutorial, we will focus on machine learning models of source code and natural language tailored to tackle these tasks. These methods have attracted wide interest not only in the NLP community, but also in software engineering and machine learning conferences, attesting to the interdisciplinary nature and broad impact of this research field. An overview of many of these methods can be found in co-proposer Allamanis' survey on the topic [1].

First, we will analyze the relationship between source code and natural language text. The similarities and differences between those types of languages drive the design of the machine learning models used for understanding and generating code and connecting it to natural language. Furthermore, despite the formal nature of programming languages, there is an abundance of natural language artifacts embedded within source code and vica-versa. We will discuss these artifacts, their special characteristics and how they relate with existing NLP research. We will also show some examples of how source code and source code repositories present an interesting type of grounding for natural language, particularly instructional or procedural language.

The remainder of the tutorial will cover specific recent methods that have been used to tackle each of the four tasks above. In doing so, we will stress a number of aspects of the presented methods:

1. Why the natural language artifacts occuring in each of the projects are interesting, and perhaps unique, compared to other sources of natural language data.
2. How to make connections between natural language artifacts and the corresponding code, and how these connections can be used to benefit each of the tasks.
3. Specific modelling techniques that have proven useful in these tasks, and how they may be fed back to other applications in mainstream NLP research.

Finally, we will close our tutorial by discussing open problems and challenges.

Outline of Contents

We aim for a three-hour tutorial to cover a reasonable range of aspects of this area. Times are approximate, and will be adjusted somewhat as we refine the tutorial content.

Part 1

- Introduction (30 minutes)
    - Motivation for modelling source code and natural language
    - Where does language appear in code and vice-versa?
    - The similarities and differences between natural and programming languages.
- Data sources (10 minutes)
- Methods for mapping from code to natural language (40 minutes)

Part 2

- Methods for mapping from language to code (45 minutes)
- Modelling natural language aspects of source code (15 minutes)
- Modelling communicative aspects of software projects (15 minutes)
- Conclusion (5 minutes)
    - Where should I start?

Names/Affiliations

Graham Neubig (gneubig@cs.cmu.edu, http://phontron.com) is an assistant professor at Carnegie Mellon University specializing in natural language processing and machine learning. One of his major research interests is models that link together natural language and code, including summarizing the intent of code in natural language, generating code from natural language, or discovering the correspondences between the two modalities. He has previously given well-attended tutorials at NLP conferences (EMNLP and YRSNLP) and the Lisbon Machine Learning Summer School, and has won a number of best papers (e.g. EMNLP2016 and EACL2017) and given invited talks, including an upcoming one on this topic at the AAAI Workshop on NLP for Software Engineering.

Miltos Allamanis (miallama@microsoft.com, https://miltos.allamanis.com) is a researcher at Microsoft Research, Cambridge, UK at the Deep Program Understanding project. He is researching applications of machine learning and natural language processing to software engineering and programming languages to create smart software engineering tools for developers. Miltos has published in both machine learning and software engineering conferences and is an author of a recent survey on machine learning for source code (https://ml4code.github.io). He received his PhD at the University of Edinburgh, UK advised by Dr. Charles Sutton.

Bibliography

[1] Allamanis, Miltiadis, et al. "A Survey of Machine Learning for Big Code and Naturalness." *ACM Computing Surveys.* To appear (2018).