From document to program embeddings: can distributional hypothesis really be used on programming languages?

Thibaut Martinet*[1]

[1]Laboratoire d'Informatique Fondamentale d'Orléans – Université d'Orléans, Institut National des Sciences Appliquées - Centre Val de Loire, Université d'Orléans : EA4022, Institut National des Sciences Appliquées - Centre Val de Loire – France

Résumé

Programming language processing is a field of increasing interest, as more and more models become available, either to address specific tasks or to acquire general knowledge which can then be fine-tuned on downstream tasks.
All these models are based on architectures that come from the field of natural language processing, most of them being built on the distributional hypothesis from linguistics.
Although this transition from one field to another appears to have occurred naturally, it is not so obvious to claim that this hypothesis will be appropriate for extracting semantics from programs.

In our work, we investigate to which extent a distributional hypothesis can be applied to code embedding.
To this end, we first formulate various hypotheses adapted to the specific information contained in programming languages. We then provide a framework to evaluate the effectiveness of these hypotheses through the quality of the resulting embedding spaces.
This framework is based on the doc2vec model as a generic language model, as its implementation of the original distributional hypothesis is easy to understand and to adapt to any new ones.
Among other tools, we propose a new evaluation method based on program analogies, which measures how well the models capture the underlying structure and meaning of the code.

We apply our framework to a set of (distributional) hypotheses and show that we can rule out certain hypotheses in favor of others.
Specifically, our study indicates that instruction-based hypotheses capture less semantic information than token-based ones. Furthermore, we observe that distributional hypotheses on tokens are effective in both source code, execution traces, and abstract syntax trees. Additionally, we find that the semantics captured on programs by these three hypotheses are of comparable levels and natures.

*Intervenant