University of Trieste, Italy
http://machinelearning.inginf.units.it/

Active Learning of Regular Expressions for Entity Extraction—Appendix

Individuals generated from matches

Whenever a new desired extraction *m* (i.e., a *match*) is available, the solver constructs 14 different individuals from *m*, by means of a deterministic heuristics ensuring that all these individuals extract *m*; all these individuals are then inserted in the population.

In this document we describe the procedure for constructing these 14 individuals from each match *m*.

Let *M* indicate the current set of matches, i.e., of annotated desired extractions. We generate from *M* a set of *tokens T* by means of the same procedure described in [Bartoli 2016]. Tokens are short text snippets that may be used as basic building blocks in the target regular expression. To identify candidate building blocks, a small set of predefined regular expressions is applied to all elements in M; the resulting snippets which occur in at least 80% of the elements in *M* are inserted in *T* (see the cited work for full details).

We construct two individuals with the following steps; let *K* denote a set of tokens (i.e., short text snippets) and let *C* denote a set of character classes:

1. one individual by:
    a. replacing all characters in *m* by the corresponding token in *K*;
    b. replacing each of the remaining characters by the corresponding class in *C*;
    c. characters that cannot be replaced are left unchanged.
2. one individual like the previous one, except that repeated terminals are compacted with the ●++ quantifier.

We repeat these steps 3 times, varying the composition of *C* and *K* as follows:

1. *C* = { `\w` } and *K* = *T*;
2. *C* = { `[A-Za-z]` , `\d` } and *K* = *T* as described above;
3. *C* = { `\w` } and K = ∅

We construct further 8 individuals with a notion of “context”---i.e., individuals with lookarounds operators.

We construct two individuals with the following steps;

1. one individual as in step 1 above, with the following difference: the individual is surrounded by a lookbehind operator filled with the 10 characters that precede *m* in the input text and by a lookaround operator filled with the 10 characters that follow *m*. If no characters precede/follow *m*, we omit the lookbehind/lookahead operator.
2. one individual like the previous one, except that repeated terminals are compacted with the ●++ quantifier and the ●{●,●}+ quantifier (if repetitions are in a lookbehind).

We repeat these steps 4 times, varying the composition of *C* and *K* as follows:

1. *C* = { `\w` } and *K* = *T*;
2. *C* = { `[A-Za-z]` , `\d` } and *K* = *T*;
3. *C* = { `\w` } and *K* = ∅
4. *C* = ∅ and *K* = *T*;

References

[Bartoli 2016] “*Inference of regular expressions for text extraction from examples*”, Bartoli A., De Lorenzo A., Medvet E., Tarlao F., IEEE Transactions on Knowledge and Data Engineering, 2016.