Inferring Method Specifications from Natural Language API Descriptions

Rahul Pandita, Xusheng Xiao, Hao Zhong, Tao Xie, Stephen Oney, and Amit Paradkar

Jiang Shun, Zhongyue Huang

Research Problem

- API documents describe legal usage of reusable software libraries
- Developers often overlook some documents and build software systems that are inconsistent with the legal usage of libraries.
- How can we solve this problem?

Code contracts

- A popular way of formalizing method specifications
- Capture pre-condition and post-condition
- **Problem**: code contracts don't exist in a formalized form in most existing software system

API documents

- Commonly existed and used in software systems
- **Problem**: documents are written in natural language, no existing tools can verify legal usage and it is time consuming and labor intensive to write code contracts manually

Objective: inferring method specifications from API documents

Motivation

```
public void service(HttpServletRequest request, HttpServletResponse response) throws
IOException{

    response.setContentType("text/plain");
    response.setHeader("Content-Disposition", "attachment;filename=sample.txt");
    ServletContext ctx = getServletContext();
    InputStream is = ctx.getResourceAsStream("sample.txt");

    int read=0;
    byte[] bytes = new byte[BYTES_DOWNLOAD];
    OutputStream os = response.getOutputStream();

    while((read = is.read(bytes))!= -1){
        os.write(bytes, 0, read);
    }
    os.flush();
    os.close();
}
```

java.lang.NullPointerException

getResourceAsStream():

"This method returns null if no resource exists at the specified path."

Insight of Approach

"Inferring code contracts from method descriptions in API documents by applying Natural Language Processing"

Challenges

"true if path is an absolute path; otherwise false"

Ambiguity

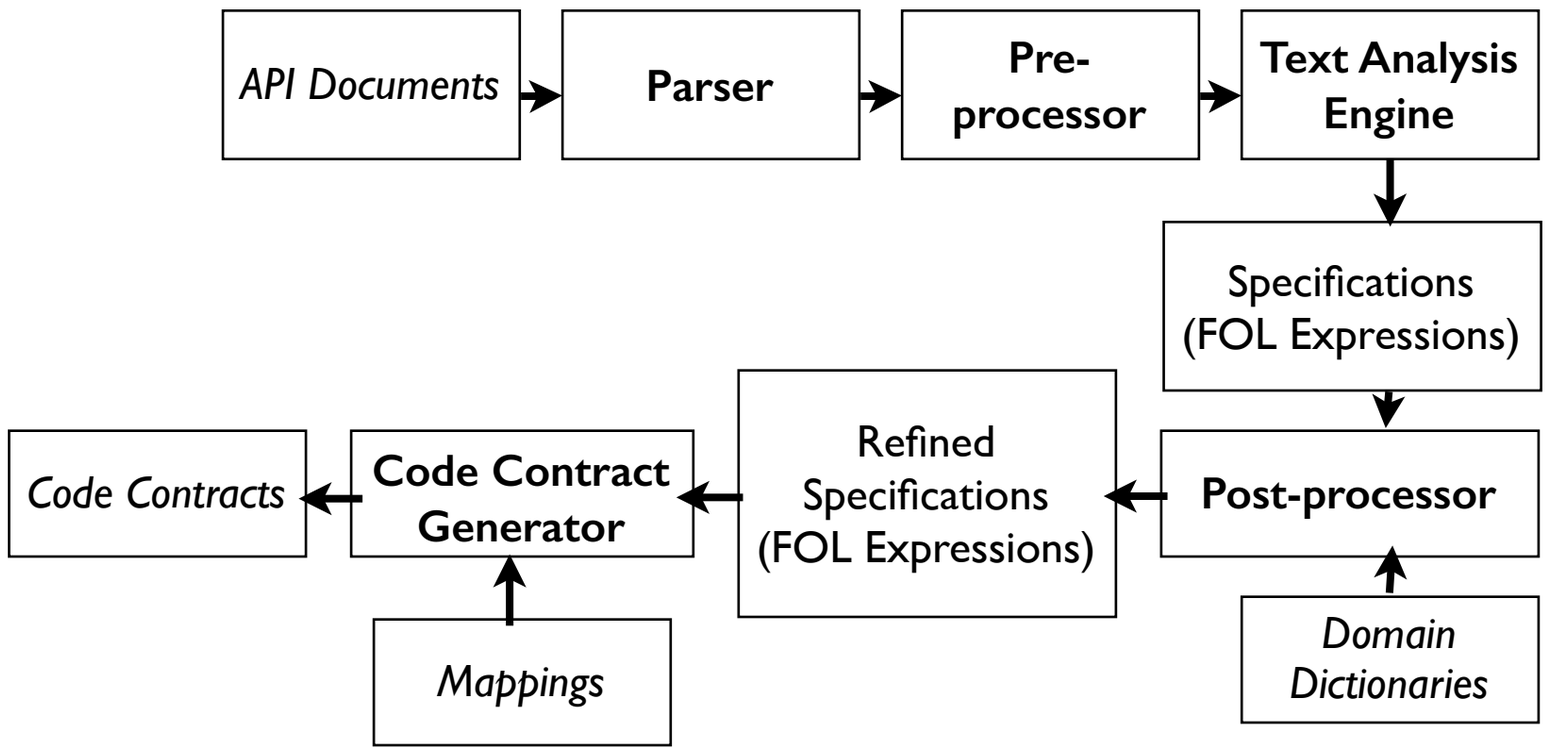
"This method also returns false if path is null"

Programming Keywords

"name can contain numbers, underscores..." and "name consists of numbers and/or underscores"

Semantic Equivalence

Overview of Approach



Parser

Extracts intermediate contents from the method descriptions of API documents

summary, argument, return, exception and remark descriptions

Pre-processor

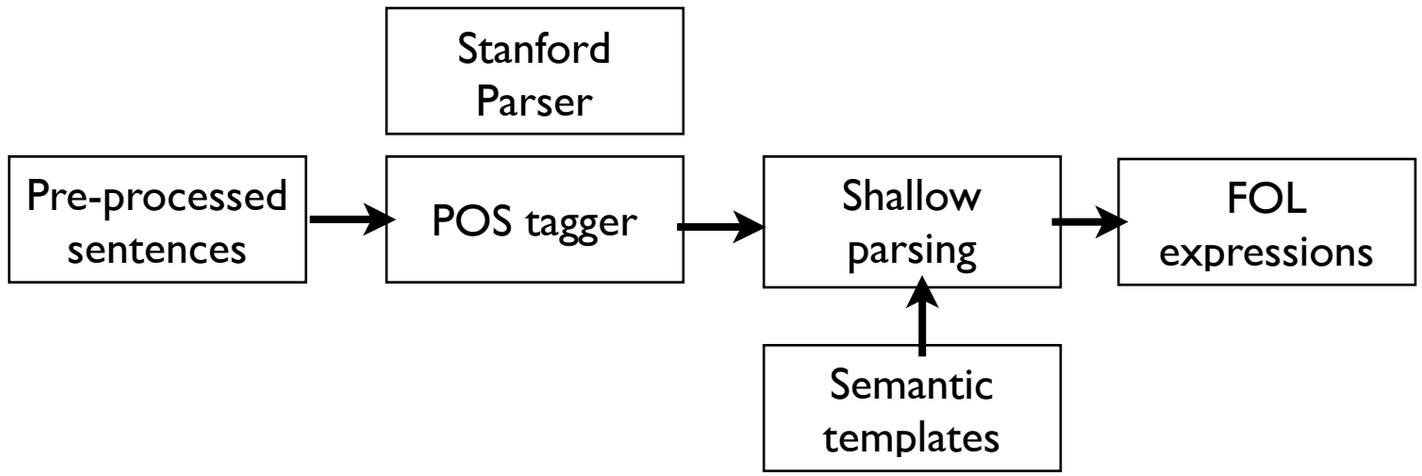
- Meta-data augmentation: names/types of arguments, types of return value/exceptions, names of classes/namespaces/ methods e.g. <param name=``prop_name``>...</param>

- Noun boosting: resolve “Program keywords” challenge by a domain specific dictionary e.g. null -> noun

- Programming constructs and jargon handling: increasing the accuracy of the POS tagger e.g. Facebook.Data -> Facebook_Data; max->maximum

Text Analysis Engine

Parses pre-processed sentences and builds specifications in the form of First Order Logic (FOL) expressions



$\text{The (path)}_{\text{subject}}\ \text{(can not be)}_{\text{verb}}\ \text{null}_{\text{object}}$

Post-processor

- Equivalence analysis: classify predicates by Lemmatization (WordNet)
  e.g. am, are and is -> be

- Intermediate term elimination: remove irrelevant modifiers
  e.g. $(\text{name})_{\text{subject}}$ $(\text{is})_{\text{verb}}$ a $(\text{valid identifier})_{\text{object-subject}}$, which $(\text{is no longer than 32 characters})_{\text{clause}}$

- Expression augmentation: augment not well written expressions
  e.g. If path is null.

Code Contract Generator

Generator uses the predefined mapping of semantic classes of the predicates to the programming constructs to produce valid code contracts

Mapping relations: String class, Integer class, null checks, return and throws constructs

“Greater ” -> length method in String class -> Requires(!name.length() > 32)

Evaluation

- C# File System API documents (File, Path and Directory)
- Facebook API documents (Data, Friends, Events, and Comments)

Evaluation

- RQ1: What are the precision and recall of the approach identifying contract sentences?
  Precision: 91.8%, Recall: 93% and F-score: 92.4% over 2717 sentences
- RQ2: What is the accuracy of the approach in inferring specifications from contract sentences in the API documents?
  Accuracy: 83.4% over 1600 contact sentences
- RQ3: How do the specifications inferred by the approach compare with the human written code contracts?
  21 in common

Results

Class [API	#M	#S	S_C	TP	FP	FN	**P**	**R**	**F_S**	S_I	**Acc**	S_D	**C**	Q
Data[Facebook.Rest]	133	810	320	288	55	32	84	90	86.9	244	76.3	102	21	0.75
Friends[Facebook.Rest]	37	215	126	96	10	30	90.6	76.3	82.8	84	66.7	17	0	0.83
Events[Facebook.Rest]	29	194	122	110	12	12	90.2	90.2	90.2	84	68.9	15	0	0.85
Comments[Facebook.Rest]	16	96	33	33	19	0	63.5	100	77.7	28	84.9	12	0	0.7
File[System.IO(.NET)]	56	795	647	627	15	20	97.7	97	97.3	599	92.6	NA	NA	NA
Path[System.IO(.NET)]	18	99	63	48	11	15	81.4	76.2	78.7	44	69.8	NA	NA	NA
Directory[System.IO(.NET)]	44	508	380	371	18	9	95.4	97.6	96.5	327	86.1	NA	NA	NA
Total	333	2717	1691	1573	140	118	**91.8**	**93**	**92.4**	1410	**83.4**	146	**21**	0.79

Conclusion

- First approach analyzes API documents to extracts specifications targeted towards generating code contracts
- The evaluation results show that the approach effectively identifies contract sentences with an average of 92% precision and 93% recall, and infers specifications from around 1700 contract sentences with an average accuracy of 83%.