Regex Feature Use In Practice

Carl Chapman and Kathryn T. Stolee
Department of Computer Science
Iowa State University
{carl1978, kstolee}@iastate.edu

***Abstract*—Regular expressions are used frequently in many programming languages for form validation, ad-hoc file searches, and simple parsing. Many researchers have created solvers, parsers, tools and theoretical works which include a subset of the regular expression features used in practice. Yet, there does not exist an empirical study of regular expression feature usage in practice that could inform the choices that researchers are making about what features to include and exclude. In this paper, we explore feature usage in practice, focusing on how often certain excluded features are used, and what use cases are associated with what features. To do this, we analyzed about 4000 open source Python projects from GitHub and explored the regular expressions contained within. Our results indicate that TODO: high level results**

I. Introduction

Regular expressions are used extensively in many programming languages, for example, to search text files [3], in form validation, and for XYZ.

II. Motivation

Bugs related to regular expressions are common, resulting in tens of thousands of bug reports [4].

III. Related Work

A. *Research on Regular Expressions*

Visual debugging of regular expressions [1]

Static analysis to reduce errors in building regular expressions by using a type system to identify errors like `PatternSyntaxExceptions` and `IndexOutOfBoundsExceptions` at compile time [4].

B. *Research on Regular Expressions*

Visual debugging of regular expressions [1]

C. *Research that Depends on Regular Expression Usage*

Regular expressions are used as queries in a data mining framework [2]

IV. Study

A. *RECORDING REGEX USAGES*

Using GHTorrent, we found the clone urls of 42,000 github projects that had Python listed as the main language. After consulting with github about the polite way to mine their service, we used 3 separate machines to clone projects in parallel. On each machine, one attempt was made to clone each project into a unique directory. 19 of these attempts failed, so a total of 3,898 projects were scanned overall. For each of these projects, the java program launched a Python process that used 'Astroid' to build the AST of each python file in the unique directory, recording uses of the 're' module. Here is an example Python code using the 're' module:

Placeholder for several examples of useage of the 're' module...image? Shows example of function, flags pattern.

TODO...mention rewinding, duplicate skipping, exact criteria for citing a usage...For each of the 53,894 regex usages observed, we recorded which 're' function was used, what flags (if any) were used, and what pattern was found.

B. *FILTERING REGEX USAGES*

Because we want to let regex patterns alone define behavior later in our analysis, all regex usages where behavioral flags are used are discarded, along with all invalid pattern strings. TODO - change to percents 6,835 usages were discarded because they used behavior-altering flags, and 3,526 without flags were discarded because their pattern strings were invalid (could not be compiled by Python into a regex object).

43,525 regex usages remain (meow percent), which are then condensed to 14,113 distinct pattern strings. The number of distinct pattern strings is much smaller because the same pattern string is often used in many distinct files and projects by different functions with different flags and these all count as separate usages. The resulting set of patten strings were parsed using TODO - [1] PCRE parser. 0.7% of the strings caused the PCRE parser to raise an error. Another 0.2% used regex features that we have chosen to exclude in this study (most notably named capturing groups). The 13,912 distinct pattern strings that remain are given a weight based upon how many projects the pattern appeared in (FYI no forked projects were scanned).

TODO - clean this: So then we use the following 4? techniques to try and best represent the entire corpus using a few regexes: 1. weight (frequency of usage across projects) 2. features (matching properties of feature usage) 3. syntactic clustering 4. semantic clustering ...and also we give the topN clones by weight, and a list that tries to include information from all 4 lists? That will be all. Future work: mine more projects and compare that result with this result. Compare across languages (java, javascript,ruby,etc.).

TODO - list research questions (metrics and how they are computed?)

RQ1: How is the `re module` used in python projects?
RQ2: How frequently are regexes used in python projects?

[1]www.source

RQ3: Which regex language features are used most commonly in python? RQ4: How syntactically diverse are regexes used in python? RQ5: How semangically diverse are regexes used in python?

1) FEATURES
Here is a table showing all the features included in this study...

code	description	example usage
DEC	any of: 0123456789	`\d`
NDEC	any non-decimal	`\D`
WSP	`\t\n\r\v\f or space`	`\s`
NWSP	non-whitespace	`\S`
WRD	letters, underscore, digits	`\w`
NWRD	non-word chars	`\W`
RNG	use of range shorthand	`[a-z]`
LKA	matching sequence follows	`a(?=bc)`
NLKA	sequence doesn't follow	`a(?!yz)`
LKB	matching sequence precedes	`(?<=a)bc`
NLKB	sequence doesn't precede	`(?<!x)yz`
NCG	group without capturing	`a(?:b)c`
CCC	custom char class	`[aeiou]`
CG	capturing group	`(caught)`
ANY	match any char	`.`
LIT	a literal char	`a`
NCCC	negated custom char class	`[^qxf]`
BKR	match the i^{th} capture	`(.)\1`
OR	logical or	`a|b`
END	match end-of-string	`$`
STR	match start-of-string	`^`
WNW	word/non-word boundary	`\b`
NWNW	negated WNW	`\B`
ADD	one-or-more repetition	`z+`
DBB	$n \leq x \leq m$ repetition	`z{3,8}`
KLE	zero-or-more repetition	`.*`
LWB	at least n repetitions	`z{15,}`
QST	zero-or-one repetitions	`z?`
SNG	exactly n repetitions	`z{8}`
ADDL	lazy ADD	`z+?`
DBBL	lazy DBB	`z{5,9}?`
KLEL	lazy KLE	`.*?`
LWBL	lazy LWB	`z{2,}?`
QSTL	lazy QST	`z??`
SNGL	lazy SNG	`z{6}?`

TABLE III. which features are studied, and how this study names them

V. Results

A. Context and Corpus Origin

1) SATURATION :
Although 42.2% of the projects observed had at least one regex usage, only 11.2% of the files observed had at least one regex usage.

From the above figure/table, we see that on average each project had 2 files containing any regex usage, out of an average of 6 files. Each of the files that did have a regex usage had an average of 1 regex usages. Because we scanned 3,898 projects, we would expect to have seen 23,388 regex usages, which is lower than the actual 53,894 usages observed.

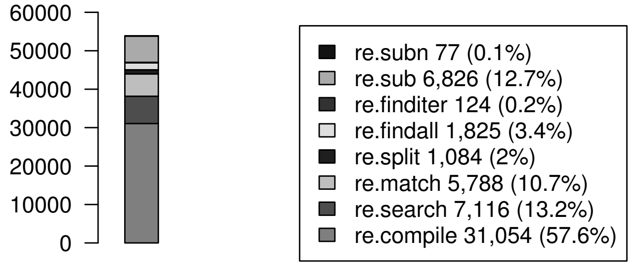

Fig. 1. How often are the 8 re functions used?

2) FUNCTIONS AND FLAGS :
As seen in Figure 1 The 'compile' function encompasses 57.6% of all usages, even though every compiled regex object can only be used by calling other functions. (TODO-Why?)

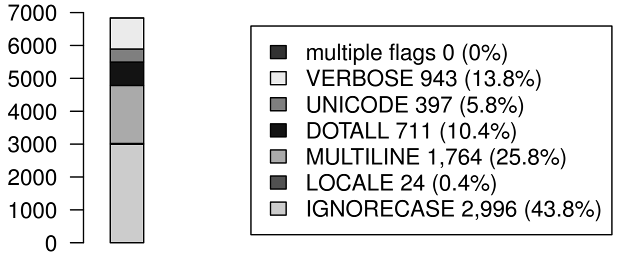

Fig. 2. Which behavioral flags are used?

87.3% of all regex usages did not use a flag or specified a non-behavioral flag (default or debug). Of all behavioral flags used, ignorecase (43.8%) and multiline (25.8%) were the most frequently used. It is also worth noting that although multiple flags can be combined using a bitwise or, this was never observed. (remove this last part if it is observed later)

3) GENERAL CHARACTERISTICS OF REGEXES FOUND :
...TODO

4) Top 10 Regex Patterns by weight :

5) All Features :
Literal tokens were found in (TODO) 101% of patterns, and accounted for 75% of all tokens. Excluding literal tokens and features that were not present in any pattern, the following stats...make a sentence, these are some stats about the features:

pattern	weight
'\\s+'	181
'\\s'	78
'\\d+'	70
'[\\x80-\\xff]'	69
'\nmd5_data = {\n([^}]+)}'	69
'\\\\(.)'	67
'([\\\\"]|[^\\ -~])'	66
'(-?(?:0|[1-9]\\d*))(\\.\\d+)?([eE][-+]?\\d+)?'	61
'[^]]+?\\] +([0-9.]+): (\\w+) <-(\\w+)'	60
'.*rlen=([0-9]+)'	57

some more text, IDK

feature	% present	% tokens	Avg	Max
DEC	17.1	5.1	2	24
NDEC	0.3	0.1	1	6
NWSP	3.5	0.7	1	10
NWRD	0.7	0.2	2	6
RNG	19.4	8.3	3	50
WSP	20.9	6.3	2	32
WRD	10.5	2.1	1	13
LKA	0.8	0.1	1	4
NLKA	1	0.2	1	3
LKB	0.6	0.1	1	4
NLKB	0.7	0.1	1	4
NCG	5.8	1.5	2	28
CCC	32.9	8.4	2	42
CG	52.1	13	2	17
ANY	33.8	7.3	2	60
NCCC	14.2	2.8	1	15
BKR	0.4	0.1	1	4
OR	15.5	2.7	1	15
END	23.4	3.4	1	12
NWNW	0	0	1	2
STR	26.3	3.8	1	12
WNW	1.8	0.4	2	36
ADD	44	11.4	2	30
DBB	2.8	0.6	2	11
KLE	43.9	11.9	2	50
LWB	0.7	0.1	1	3
QST	13.8	3.4	2	35
SNG	4.5	1.3	2	17
LZY	9.5	1.8	1	12
NCG	6.7	2.5	3	16
ENDZ	0.7	0.1	1	1
VWSP	0.1	0	1	2
OPT	1.7	0.2	1	2
references named capture group	0.1	0	1	2

pair	example from corpus	nTimes
CG::ADD	'(:+)'	4189
CG::KLE	'(:)*'	3983
ANY::KLE	'.*'	3709
CG::ANY	'(.)'	3160
CCC::CG	"(['])"	2665
CCC::ADD	'[ ]+'	2612
RNG::CCC	'[A-Z]'	2567
ADD::KLE	'-*(.+)'	2476
WSP::KLE	'\\s*'	2207
END::STR	'^$'	2156

OK now that is all for section 2. Now in section 3 I want to look at clustering by string similarity using mcl clustering algorithm. Here are the top 6 clusters using various string similarity metrics:

TODO - multiple boxplots for all 5-6 demonstrating cluster size and then also have # of clusters, pick smallest number of clusters and then use that.

VI. Discussion

...only 11.2% of the files observed had at least one regex usage. This indicates that regex usage may usually be concentrated in just a few files.

VII. Conclusion

Acknowledgment

This work is supported in part by NSF SHF-1218265, NSF SHF-EAGER-1446932, and the Harpole-Pentair endowment at Iowa State University.

References

[1] F. Beck, S. Gulan, B. Biegel, S. Baltes, and D. Weiskopf. Regviz: Visual debugging of regular expressions. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 504–507, New York, NY, USA, 2014. ACM.

[2] A. Begel, Y. P. Khoo, and T. Zimmermann. Codebook: Discovering and exploiting relationships in software repositories. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 125–134, New York, NY, USA, 2010. ACM.

[3] C. L. A. Clarke and G. V. Cormack. On the use of regular expressions for searching text. *ACM Trans. Program. Lang. Syst.*, 19(3):413–426, May 1997.

[4] E. Spishak, W. Dietl, and M. D. Ernst. A type system for regular expressions. In *Proceedings of the 14th Workshop on Formal Techniques for Java-like Programs*, FTfJP '12, pages 20–26, New York, NY, USA, 2012. ACM.