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 and which features are supported by four popular regex research projects/tools:

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.

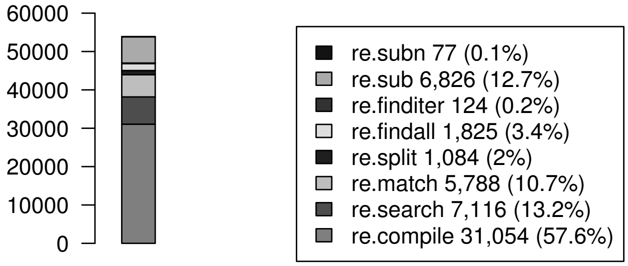

Fig. 1. How often are the 8 re functions used?

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?)

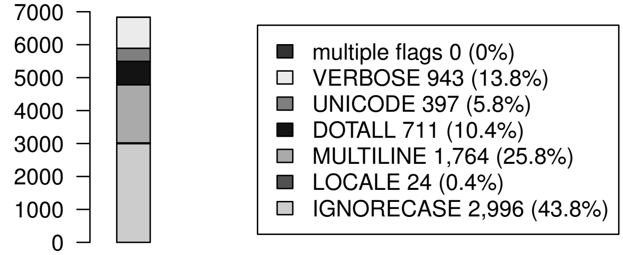

Fig. 2. Which behavioral flags are used?

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:

some more text, IDK

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.

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

code	description	example	brics	hampi	Rex	RE2	% tokens	Max	weight	weight%
CG	a capture group	(caught)	●	●	●	●	13	17	16,890	47.8
KLE	zero-or-more repetition	.*	●	●	●	●	11.9	50	14,767	41.8
ADD	one-or-more repetition	z+	●	●	●	●	11.4	30	14,379	40.7
CCC	custom character class	[aeiou]	●	●	●	●	8.4	42	11,558	32.7
ANY	any non-newline char	.	●	●	●	●	7.3	60	9,901	28.0
STR	start-of-line	^	○	●	●	●	3.8	12	9,153	25.9
END	end-of-line	$	○	●	●	●	3.4	12	8,411	23.8
WSP	\t \n \r \b \f or space	\s	○	●	●	●	6.3	32	7,751	21.9
RNG	chars within a range	[a-z]	●	●	●	●	8.3	50	6,901	19.5
DEC	any of: 0123456789	\d	○	●	●	●	5.1	24	5,630	15.9
OR	logical or	a|b	●	●	●	●	2.7	15	5,498	15.6
NCCC	negated CCC	[^qwxf]	●	●	●	●	2.8	15	5,293	15.0
QST	zero-or-one repetition	z?	●	●	●	●	3.4	35	4,958	14.0
WRD	[a-zA-Z0-9_]	\w	○	●	●	●	2.1	13	3,861	10.9
LZY	as few reps as possible	z+?	○	●	○	●	1.8	12	3,119	08.8
NCG	group without capturing	a(?:b)c	○	●	○	●	1.5	28	2,316	06.6
NCG	named capture group	(?P<name>x)	○	●	○	●	2.5	16	2,078	05.9
SNG	exactly n repetition	z{8}	●	●	●	●	1.3	17	1,562	04.4
NWSP	any non-whitespace	\S	○	●	●	●	0.7	10	949	02.7
DBB	$n \leq x \leq m$ repetition	z{3,8}	●	●	●	●	0.6	11	874	02.5
WNW	word/non-word boundary	\b	○	○	○	●	0.4	36	643	01.8
NLKA	sequence doesn't follow	a(?!yz)	○	○	●	○	0.2	3	582	01.6
OPT	options wrapper	(?i)CasE	○	●	○	●	0.2	2	573	01.6
LKA	matching sequence follows	a(?=bc)	○	○	○	○	0.1	4	401	01.1
NLKB	sequence doesn't precede	(?<!x)yz	○	○	○	○	0.1	4	355	01.0
LWB	at least n repetition	z{15,}	●	●	●	●	0.1	3	316	00.9
NWRD	non-word chars	\W	○	●	●	●	0.2	6	288	00.8
LKB	matching sequence precedes	(?<=a)bc	○	○	○	○	0.1	4	268	00.8
BKR	match the i^{th} CG	\1	○	○	○	○	0.1	4	213	00.6
ENDZ	absolute end of string	\Z	○	○	○	●	0.1	1	172	00.5
NDEC	any non-decimal	\D	○	●	●	●	0.1	6	83	00.2
BKRN	references NCG	\g<name>	○	●	○	○	0	2	44	00.1
VWSP	matches U+000B	\v	○	○	●	●	0	2	23	00.1
NWNW	negated WNW	\B	○	○	○	●	0	2	11	00.0

[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.