Boa: A Link Between Worlds

Stephen Romansky[1], **Sadegh Charmchi**[1], **and Abram Hindle**[1]

[1]**Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada,**

ABSTRACT

The business models of software/platform as a service have contributed to developers dependence on the Internet. Developers can rapidly point each other and consumers to the newest software changes with the power of the hyper link. But, developers are not limited to referencing software changes to one another through the web. Other shared hypermedia might include links to: Stack Overflow, Twitter, and issue trackers.

This work explores the software traceability of Uniform Resource Locators (URLs) which software developers leave in commit messages and software repositories. URLs are easily extracted from commit messages and source code. Therefore, it would be useful to researchers if URLs provide additional insight on project development.

To assess traceability, manual topic labelling is evaluated against automated topic labelling on URL data sets. This work also shows differences between URL data collected from commit messages versus URL data collected from source code. As well, this work explores outlying software projects with many URLs in case these projects do not provide meaningful software relationship information. Results from manual topic labelling show promise under evaluation while automated topic labelling did not yield precise topics. Further investigation of manual and automated topic analysis would be useful.

Keywords: Software Engineering, Traceability, Source Code, Commit Message

1 INTRODUCTION

Software services and platforms have been thriving in recent years due to their affordability for small businesses and consumers. Instead of having to purchase your own enterprise-grade server equipment, and having to administer its setup, you can rent just the right amount of resources you need from a service company to host your web services. Software services can provide web interfaces that clients of the service may interact with. Hence, it is becoming more common for web service software to provide locators (URLs) to serve content and receive instructions from clients.

GitHub and SourceForge provide version control systems (VCSs) which act as software as a service applications. Software developers can take advantage of VCS services to help with reliably storing software changes during development. VCS alleviates the worry of deleting important code and it enables developers to more easily share their software changes with whomever is interested in them. It is also possible for software developers to increase their code traceability by associating their development code on a VCS with tools like issue trackers. Developers can manually add links between software and software artifacts like mail, bug trackers, or code documentation.

The purpose of our work is to find out the tools developers use such as: web forums, social media, domain specific references, or variations of development tool configurations. Our work takes advantage of the software repository data curated by the Boa software project Dyer et al. (2013). We pick a subset of Boa GitHub software projects and all Boa commit messages available for GitHub related projects for extracting URLs. Our work checks if there is a difference between URLs which are embedded in commit messages compared to URLs which are embedded in source code. To accomplish these tasks, commit data is collected and parsed using the Boa platform while source code and source code changes can be copied from GitHub given the project URL from Boa. It is then possible to query these URLs for whether or not a software project uses software development tools like issue, bug or build tracking tools in their software development process. This work explores whether or not parsing URLs provides meaningful information about software development process used by software projects.

2 RELATED WORK

Kalliamvakou *et al.* Kalliamvakou et al. (2014) have investigated GitHub for properties of projects hosted on the website. The properties which are particularly relevant to this work: most projects have few commits, repositories are not guaranteed to be software projects, and that many active projects do not work only using GitHub. Barua *et al.* Barua et al. (2014) use latent Dirichlet allocation (LDA) to extract topics from software questions and answers on the developer website Stack Overflow. Hindle *et al.* Hindle et al. (2012) use LDA to extract software requirement traceability data. Our work applies LDA for topic-based software hyperlink traceability information.

Keivanloo and Rilling Keivanloo and Rilling (2014) construct a set of semantic graph links between software code using code analysis and provide a means of sharing this linked data by beginning the construction of an open data analysis tool. Keivanloo and Rilling do not look at expanding their set of projects through URLs like we propose. Therefore, our work is novel and could be applicable to their semantic security web. Dyer *et al.* Dyer et al. (2013) is relevant to our work as well as it provides a corpus of software repositories for our investigation and methods for analyzing commit messages. No prior work appears to investigate hyperlink traceability from source code and commit messages.

3 METHOD

To evaluate topic extraction from URLs several data processing steps are taken. Data is collected using Boa to return all commit messages associated with Boa's GitHub dataset and the GitHub project repository URLs. A subset of the GitHub projects are then copied from GitHub onto a researchers machine. A program was written and used to parse each collected commit message for URLs. The same program was modified to iterate through each copied GitHub project and analyze the projects changesets for added or modified URLs. The commit URL and source code URL datasets were then tokenized and assessed with expert crafted labels. LDA was applied to the source code URL dataset to evaluate unsupervised topic labelling.

The commit messages and source code URLs were tokenized using the following delimiters:

```
/,=,?,:,#,&
```

Tokenization of the URLs was chosen to determine if frequently appearing tokens had immediate semantic meaning with respect to software development. For instance, the word "bug", "issue", or "docs" appearing in a URL are usually associated with software development services. Our work explores taking advantage of these simple word-software conventions by trying to label projects based on the URLs found in the projects commit messages and source code.

To collect URLs out of source code history a program was written to replay commits over time and to calculate the code difference between consecutive commits. Lines that were added by new files, or modified from a change to an existing file, were recorded and processed for URLs. Each mined project was then associated with a set of commits, and the commits with a corresponding set of URLs depending on whether or not the software changes contained detected URL changes.

To generate topic labels for the commit message URLs, the 500 most frequently occurring tokens were analyzed by one of the authors. Such tokens include: repo, wiki, doc, developer, jira, mailman, gerrit, png, twitter, ticket, and issue. These tokens were grouped into the categories: repository, documentation, development tools, and communication depending on what the URL token is referring to as approximated by the author. These tokens are significant with respect to software process. For instance, jira and gerrit provide features for reviewing code changes developers are making. This allows development teams to create higher quality code, catch bugs, and audit their software. Tokens like twitter and mailman are likely to contain social network information relevant to the software project like how open source developers and software users are communicating with one another.

Table 1 shows a list of manually extracted categories and a subset of the keywords used to generate each category. The category keywords can be applied to tokens from a URL to approximate which part of the software process the URL is associated with. To simplify data comparison the label categories were merged into 4 topics. Table 3 shows counts for these labelled groups from the git commit message data. The *repository* label is attached to URLs that contain tokens related to repositories; the *documentation* label is attached to URLs that contain tokens related to development documentation like platform information or API information; the *development tools* label is used to refer to software services like issue trackers,

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1947v1 | CC-BY 4.0 Open Access | rec: 11 Apr 2016, publ: 11 Apr 2016

Table 1. Manual Topic Labelling and Topic Keywords

Category	Keywords
repository	svn, git, hg, repo
forum	thread, post, forum, group
document	doc, dev, wiki, lib
issue	ticket, issue, tracker, bug
dev tools	jira, gerrit, review
media	twitter, blog, articles, png, gif
communication	mail, stackoverflow

Table 2. Counts of Protocol Accessing GitHub

Protocol	Commits	Percent of Commits
http	11 734	3
https	217 636	69
ssh	38 428	12
git	19 949	6
other	27 806	8

code review, and data sharing; and the *communication* label is used to identify URLs that lead to blogs, forums, social media, images, or mailing lists.

For automated topic extraction LDA is applied to the tokenized URLs from source code. The URLs are grouped by project and separately by commit message. LDA is used to generate different numbers of topics. In this work the number of extracted topics investigated are: 10, 20, 30, and 50. The extracted topics can then be given a label by the researcher based on the highest weighted URL tokens found in each topic.

4 RESULTS

4.1 Commit Message Topic Labelling

4 399 711 URLs were extracted from 23 229 427 commit messages from 380 125 GitHub projects or 0.19 URLs per commit message. It was found that most of the commit URLs were created by the tool `git-svn` which synchronizes `git` and `svn` projects. 81.9% of the commit message URLs contained the word "svn" and referenced a project and software revision, due to the format of svn commits. Only 7.17% of the commit message URLs contained the word "git". The majority of commit messages containing "git" pointed to software repositories on GitHub. Table 2 shows the prefixes used on git related commit messages.

From the preliminary analysis of checking which URLs contain "git" or "svn" it is not surprising that the number of repository labels is so high in Table 3. This result does show that most of the URLs from commit messages do not contain references to external projects. This is likely a result of sample bias due to several of the projects using git-svn which creates many uninformative URLs in commit messages.

The other labelled commit message URLs, from Table 3, are still useful in that: the URLs could be used to label what type of project is under development based on documentation references; whether or not, the software project is using some form of code review, continuous integration, or issue tracking service; and, if the project is using a forum or other web service communication tools.

4.2 Manual Source Code Topic Labelling

Boa does not provide source code corresponding to the repositories it has collected data on. Therefore, this work tried to reproduce 30 000 of the 380 125 reported GitHub repositories. However, after trying to reproduce 29 908 repositories it was found that only 26 417 could be successfully retrieved. The repositories used 588 gigabytes of storage which more than expected. Therefore, the first 3 739 repositories were used in URL extraction due to time constraints on URL extraction. The 3 739 repositories were chosen by lexicographically sorting the Boa project ids. This is a threat to validity since random sampling

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1947v1 | CC-BY 4.0 Open Access | rec: 11 Apr 2016, publ: 11 Apr 2016

Table 3. Commit Message URL Topic Counts

Category	Number of URLs	Percent of URLs
Repository	3 996 116	90
Documentation	226 889	5
Development Tools	38 973	0
Communication	19 298	0
Other	118 435	2

Table 4. Source Code URL Topic Counts

Category	Number of URLs	Percent of URLs
Repository	190 058	2
Documentation	510 770	7
Development Tools	14 835	0
Communication	436 344	6
Other	5 785 093	83

Table 5. URL Schemes from Source Code

Scheme	Count	Scheme	Count
rsync	42	sip	1 093
rtsp	221	sips	20
ftp	6 236	ldap	299
data	30 724	git	2 611
https	459 585	news	262
mailto	18 684	mms	13
pop	246	svn	38 969
nntp	15	ssh	931
snews	3	file	170 150
telnet	43	ldaps	29
http	6 195 328	urn	11 379
svn+ssh	134	gopher	73

was not used. 77 374 unique hostnames were found from URLs with http, https, or ftp schemes. Table 5 lists the URL schemes that were found in the source code.

Similarly to the commit message URLs, the source code URLs were labelled using the four categories: repository, documentation, development tools, and communication as shown in Table 4. There is a large difference between the grouping from the commit messages compared to the groupings from the source code. The number of repository related URLs has changed from 90.8% to 2.74% and the number of communication related URLs has increased from 0.44% to 6.29%. References to development tools was also smaller portion in the source code group. The change in development tools may be due to the fact that commit messages are more coupled with services like issue trackers. This shows that commit messages and source code contain different distributions of software process related URLs.

To investigate the difference between Table 3 and Table 4 source code projects with large numbers of extracted URLs are investigated. Table 6 shows a set of projects with large ratios of URLs to number of commits. The median number of URLs per number of commits is 1.4 from the mined projects. One hypothesis for the difference between the tables is that the source code contains random URL spam from web services like URL shorteners or spammers. Upon investigation it was found that out of the top 100 projects with respect to URLs/Commit, only 3 projects had more than 50 commits.

The projects from Table 6 are as follows: MuraCMS, a content management system which lets developers build and maintain websites more easily; sparqles, a sparql endpoint status monitoring tool for checking the availability of sparql resource description framework database nodes; TSA (tikal_share_android), an android application that provides end users with YouTube content from the Tikal YouTube channel; Thesis, is a thesis project which mined Twitter accounts; epidetect, is called a natural language pro-

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1947v1 | CC-BY 4.0 Open Access | rec: 11 Apr 2016, publ: 11 Apr 2016

Table 6. Examples of Outlying Projects Contributing URLs

Project	Boa id	URLs	URLs/Commits
MuraCMS	1 067 525	35 898	6
sparqles	10 820 907	242 937	725
TSA	10 888 943	11 356	203
Thesis	10 359 307	102 636	4 665
epidetect	10 720 252	56 783	591
emacs_config	10 491 974	9 784	127
sumtotal-connector	10 178 789	29 605	1 644

cessing project used to study epidemic information from text corpuses created as a postgraduate thesis project; emacs_config, is a project which contains a programmers emacs text editor configuration; and sumtotal-connector, an Internet service framework for build event driven architecture applications.

Further manual inspection showed that distributed-iris-reasoner is a resource description framework which links web resources together. Unfortunately, distributed-iris-reasoner collects nonsoftware related information and anonymizes it like the following URL: `http://www.Department0.University1.edu/FullProfessor2/`. More than 50% of the extracted source code URLs come from the distributed-iris-reasoner project. This causes a significant portion of the data to be labelled as Other in Table 4 since it is not immediately related to software development. However, not all of the large URL using projects are labelled as Other. Table 6 lists web-integrated software applications that contain relevant URLs. This URL usage is interesting because it could be used to identify web dependent software projects.

4.3 Automated Source Code Topic Labelling

LDA typically uses two parameters to control how many keywords belong to a topic and how many documents belong to a category. In this work we refer to the parameters as α and β respectively. A document in this work refers to the set of URLs grouped by commit or by project. The parameters are set as $\alpha = 0.01$ and $\beta = 0.01$ for the majority of analysis. LDA was applied to the tokenized source code URLs twice by grouping URLs by commit message and then by project which the URL occurred in. Extracted topic size was varied using topic counts of 10, 20, 30, and 50. The author attempted to label each topic given the highest weighted keywords. However, the highest weighted keywords were frequently ambiguous such as “apache”, “http”, “android”, or “amazonaws”. The lack of strongly related weighted keywords caused many of the topics to be unlabelled. In some cases related keywords occurred together but these topics were either about GitHub repositories or Java, which makes sense but is not useful.

To try and refine the data from the extracted topics, additional approaches used with LDA. One example was only using the hostnames of the URLs were input to LDA and grouping the hostnames by commit. This did not improve the keyword grouping in each of 50 topics. The second alternative approach applied was altering the α and β values to be 1 over the number of projects which is 0.0002 with 50 extracted topics and the original source code URL data. The parameter change resulted in: 5 topics being identified as the “distributed-iris-reasoner” as they contained almost entirely RDF formatted tokens, and 1 topic being identified as “amazonaws” as it contained web API response keywords and additional variations of the amazonaws token.

No decisive conclusion can be drawn from the application of LDA to tokenized source code URLs other than outlying projects can bias the results. It is also possible that topic keywords formed a group but the author did not recognize it due to how specific the tokens were. To deal with these problems, future work could hand pick a dataset of projects with co-occurring URLs to evaluate if topics can be extracted on a known dataset.

4.4 Project URL Queries

An application of having identified URL tokens which have a relationship with software development process is the matter of using these tokens to query software projects for development tools. We apply queries for software tools related to URLs on the source code data set. Table 7 shows several URL queries and the number of projects which contain them from the source code URL dataset. The *localhost* query

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1947v1 | CC-BY 4.0 Open Access | rec: 11 Apr 2016, publ: 11 Apr 2016

Table 7. Software Process Queries on Source Code URLs

Query	Number of Projects
localhost	510
bug—ticket—issue	715
jira—gerrit	139
doc.—dev.	459
ajax—jquery—angular—reactjs	377
api.	302
stackoverflow	235
travis-ci	83

informs us how many projects involve interprocess communication to a service running on the developers computer; the *bug* query informs us how many projects make references to bugs, tickets or issue trackers; the *jira* query informs us how many project are likely to be using a code review processes which allow developers or third parties to perform code inspections or audits; the *doc.* subdomain query checks for projects that are referencing documentation; the *ajax* query looks for projects that reference web related technologies; the *api.* subdomain query looks for projects which are utilizing web service apis; the *stackoverflow* query looks for projects which have taken advantage of community collaboration on Stack Overflow; and the *travis-ci* query looks for projects which use the practice of continuous integration which involves building and testing the given software project after each new changeset is added to the version control system. The *localhost* query was refined to show that 50 projects used https and 475 projects used http. This demonstrates collecting high level information quickly from software repositories and answers our first research question, can meaningful software relationships be extracted from URLs.

5 CONCLUSION

Our exploration has shown that meaningful software related tokens can be extracted from URLs. That both commit message URLs and source code URLs can refer to software process related topics as well as unrelated topics. We also looked at extracting information about the software tools GitHub projects are using by querying the URLs they contain. The biggest contribution is the demonstration of manual and automatic topic extraction from commit and source code URLs and how well manual topic labelling can provide additional insight into the development process of software projects.

REFERENCES

Barua, A., Thomas, S. W., and Hassan, A. E. (2014). What Are Developers Talking About? An Analysis of Topics and Trends in Stack Overflow. *Empirical Software Engineering*, 19(3):619–654.

Dyer, R., Nguyen, H. A., Rajan, H., and Nguyen, T. N. (2013). Boa: A Language and Infrastructure for Analyzing Ultra-Large-Scale Software Repositories. In *35th International Conference on Software Engineering*, ICSE 2013, pages 422–431.

Hindle, A., Bird, C., Zimmermann, T., and Nagappan, N. (2012). Relating requirements to implementation via topic analysis: Do topics extracted from requirements make sense to managers and developers? In *Proceedings of the 28th IEEE International Conference on Software Maintenance*. IEEE.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., and Damian, D. (2014). The promises and perils of mining github. In *Proceedings of The 11th Working Conference on Mining Software Repositories*, pages 92–101. ACM.

Keivanloo, I. and Rilling, J. (2014). Software Trustworthiness 2.0-A Semantic Web Enabled Global Source Code Analysis Approach. *Journal of Systems and Software*, 89:33–50.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1947v1 | CC-BY 4.0 Open Access | rec: 11 Apr 2016, publ: 11 Apr 2016