Makar: A Framework for Multi-source Studies based on Unstructured Data

Mathias Birrer*, Pooja Rani*, Sebastiano Panichella†, Oscar Nierstrasz*

*Software Composition Group, University of Bern
Bern, Switzerland
scg.unibe.ch/staff
† Zurich University of Applied Science (ZHAW)
panc@zhaw.ch

***Abstract*—To perform various development and maintenance tasks, developers frequently seek information on various sources such as mailing lists, Stack Overflow (SO), and Quora. Researchers analyze these sources to understand developer information needs in these tasks. However, extracting and preprocessing unstructured data from various sources, building and maintaining a reusable dataset is often a time-consuming and iterative process. Additionally, the lack of tools for automating this data analysis process complicates the task to reproduce previous results or datasets.**

To address these concerns we propose *Makar*, which provides various data extraction and preprocessing methods to support researchers in conducting reproducible multi-source studies. To evaluate Makar, we conduct a case study that analyzes code comment related discussions from SO, Quora, and mailing lists. Our results show that Makar is helpful for preparing reproducible datasets from multiple sources with little effort, and for identifying the relevant data to answer specific research questions in a shorter time compared to the manual investigation, which is of critical importance for studies based on unstructured data. Tool webpage: https://github.com/maethub/makar

***Index Terms*—Mining developer sources, Code comments, Stack Overflow, Mailing lists**

I. Introduction

As a software system continues to evolve, it becomes bigger and more complex, and developers need various kinds of information to perform activities such as adding features, or performing corrective maintenance [1]. Developers typically seek information on internal (available within IDE) or external sources such as Q&A forums,[1] Github[2] to satisfy their information needs as shown in Figure 1 [2].

To support developers in various activities and understand their information needs, researchers have analyzed these external sources such as Github, CVS, mailing lists, and CQA sites [3] (see Figure 1). However, extracting and preprocessing unstructured data from these sources, and maintaining the data due to lack of automated techniques pose various challenges in conducting reproducible studies [4], [5], [3]. To gain a deeper understanding of these challenges, we surveyed the literature that focuses on studying developers information needs from different external sources (see section II).

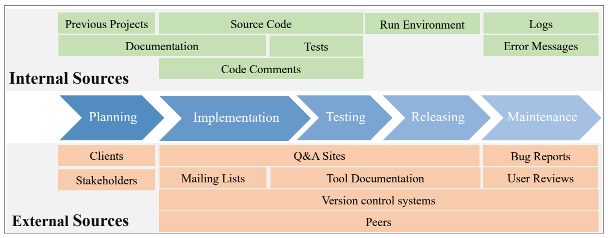


Fig. 1. Developers seek various sources during software development

Prior works have raised and identified the crucial factors affecting the reproducibility of the mining studies such as data retrieval methodology, data processing steps, or dataset availability [6], [5], [4]. Chen *et al.* pointed out that 50% of articles do not report whether word stemming, a common text preprocessing step, is used or not [4]. Amann *et al.* pointed out that only 29% of the mining studies made their dataset available [5]. As a consequence, more tools and techniques are required to enable the preprocessing and analysis of multi-source studies to facilitate their replicability.

To address these concerns, we propose *Makar*, a tool that leverages popular data retrieval, processing, and handling techniques to support researchers in conducting reproducible studies. We establish its requirements based on the surveyed studies. To evaluate Makar, we conduct a case study that analyzes code comment related discussions from SO, Quora, and mailing lists. Thus the contribution of this paper is three-fold:

- We present the challenges researchers face in mining and analyzing the unstructured data from the external sources.
- We present Makar, a tool to support researchers in conducting multi-source and reproducible empirical studies.
- We report the results of the multi-source empirical study

We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project "Agile Software Assistance" (SNSF project No. 200020-181973, Feb. 1, 2019 - April 30, 2022).

[1]www.stackoverflow.com
[2]https://github.com/

conducted using Makar.

II. Background Study

To identify the challenges researchers face with various sources, we surveyed the relevant papers considered in a recent systematic literature review (SLR) [7]. The SLR includes the studies in which researchers collected developer information needs by interviewing people (people-centric) or from online platforms (technology-centric) for the program comprehension tasks [7]. We included only technology-centric studies (29 studies) due to our interest in the external sources. Following the same inclusion and exclusion criteria from the SLR (*e.g.*, studies not older than 15 years), we further included 23 additional papers that focus on studying developer information needs from other sources such as mobile app stores (*e.g.*, user reviews) and Quora, resulting into a total of 52 papers. As we aim to focus on the diversity of sources rather than on a deep overview of a particular source, we excluded the studies analyzing the same project from the same source. The list of selected papers and detailed methodology is reported in the "Background Study" file in the replication package.[3]

Challenges in various sources. Table I reports our main findings, where the column *Source* represents the source, and the column *Major challenges* reports the major challenges associated with each source. The results show that the sources significantly differ on the kinds of data they contain, the extraction methods used, and the involved communities. While a few sources offer convenient ways for extracting data (*e.g.*, SO), there are other sources (*e.g.*, Quora) that are more prohibitive and complex to acquire the required data. Similarly, extracting and processing data from mailing lists require manual efforts. Therefore, extracting the data manually is still widely adopted in practice. However, the use of manual extraction methods can lead to inconsistent collection and processing of data across sources, which impacts the reproducibility of the studies. We report the challenges Makar addresses currently (✔), those planned for the future (FW), and those not planned (✖) in the column *Makar* of Table I.

Requirements. Based on the gathered challenges in the survey, we identified relevant functional and non-functional requirements for Makar. The tool intends to cover the common use cases found in the survey while being extensible to support additional or more specific scenarios encountered in the case study. It can also be used by developers to manage their information in development as depicted in Figure 1.

We identified five main *functional requirements*: data import, data management, data processing, data querying, and data export. *Data import* focuses on the ways to import the data into the tool, *Data management* on building and maintaining the data, *Data processing* focuses on the need to preprocess the data (HTML removal, stop word removal), *Data Querying* on searching the data, and *Data export* focuses on exporting the data from the tool in order to support further analysis. We also identified *non-functional requirements* for Makar. It should be easily extensible in areas where the projects have different technical requirements, such as import adapters, preprocessing steps, or export adapters. The tool should be able to handle large amounts of data (scale of 100k records) and still have acceptable usage performance (*e.g.*, for search queries).

[3]https://doi.org/10.5281/zenodo.4281467

III. Makar Architecture

Makar has been developed as a web application so that it can be hosted on accessible and possibly powerful servers. Thus, it allows multiple users to work with the same dataset. It is a Ruby on Rails (RoR)[4] web application with a Postgresql[5] database in the back end. To have minimal technical requirements to run the tool, to provide maximal compatibility and ease of installation on different platforms and operating system, Makar runs in a Docker container. [6] We provide instructions to run the tool on the tool repository[7] and its demonstration on Youtube.[8] We show its architecture and features in Figure 2 and the next paragraphs.

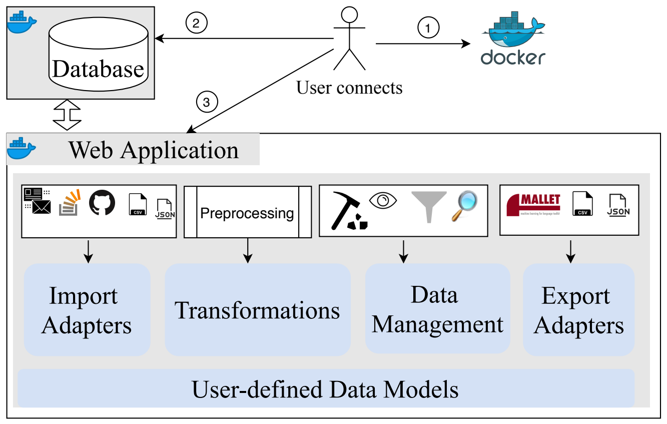


Fig. 2. Architecture overview of Makar

- **Data Import**: the user can import data from diverse sources such as *CSV* and *JSON* directly. The tool also supports direct import adapters for the following sources: *Apache Mailing List Archive*,[9] *Github Pull Requests* (via Github Archive),[10] *Github Issues* Via the Github API,[11] and *Stack Overflow* Search Excerpts.[12] The import adapters can be extended easily using `ImportAdapter` component for other sources shown in Table I
- **Data management**: Makar provides *schemas*, *collections*, *filters* and *records* to manage datasets as shown

[4]https://rubyonrails.org/
[5]https://www.postgresql.org/
[6]https://www.docker.com/
[7]https://github.com/maethub/makar
[8]https://youtu.be/Yqj1b4Bv-58
[9]https://mail-archives.apache.org/mod_mbox/
[10]http://www.gharchive.org/
[11]https://developer.github.com/v3/search/#search-issues
[12]https://api.stackexchange.com/docs/excerpt-search

TABLE I
CHALLENGES FROM EACH DATA SOURCE

Source	Major challenges	Makar
SO	- Preprocessing data for the study (*e.g.*, removing noisy data such as HTML tags, code snippets)	✔
	- Selecting relevant data (*e.g.*, the relevant and pertinent questions)	FW
Quora	- No official API available to access data and no publicly available dataset	✖
	- Finding relevant topics and questions, extracting and preprocessing data for the study	✔
Mailing lists	- The unstructured text in the mails requires manual data extraction	✔
	- Content of mails is characterized by heterogeneous types of information (stack traces, simple text, footers) and auto-generated emails from other sources requires manual analysis.	FW
Bug Reports	- Data extraction is performed manually	FW
	- Information overload and it requires human interpretation to select relevant data	✖
User Reviews	- No public API to access and extract user review data	FW
	- They require manual analysis and human interpretation to select relevant data	✖
	- Suffers from a sampling bias	✖

in Figure 2. *Schemas* define the structure of a dataset and its records, and *records* are rows of the dataset (similar to schemas and records in databases). *Collections* are arbitrary selections of records, which can be used to manage various subsets of the data. A record can belong to multiple collections. *Filters* are the search queries that help one to filter data from existing collections or schemas, and can be saved to provide efficient querying and rebuilding of the dataset. For example, a study analyzing SO questions imports the SO dataset into Makar. The study design requires only questions having the word "javadoc" in the question title. To fulfill this requirement, the user can create a filter (*e.g.*, "All Questions with Javadoc in Title" filter) by searching the question titles for "javadoc" as shown in Figure 3. The user can create a collection that uses this filter and use the collection as their dataset for further analysis. In the case, the user add more data from SO to update her dataset (collection), Makar facilitates syncing the collection using the *Auto-filter* option (reapplying the same filter) as shown in Figure 4.

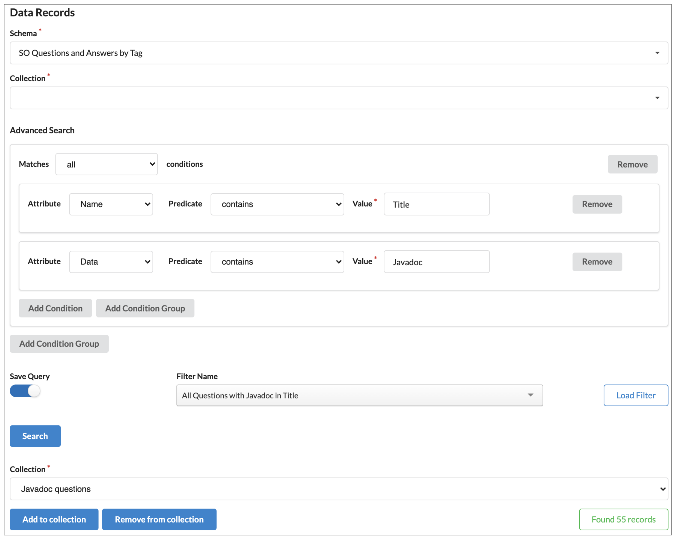

Data Records
Schema *
SO Questions and Answers by Tag
Collection *
Advanced Search
Matches: all conditions
Attribute: Name | Predicate: contains | Value *: Title
Attribute: Data | Predicate: contains | Value *: Javadoc
Save Query
Filter Name: All Questions with Javadoc in Title
Collection *: Javadoc questions
Found 55 records

Fig. 3. Search Interface of Makar

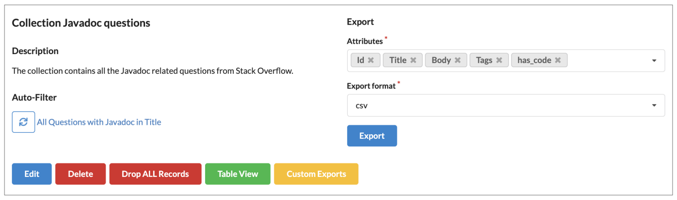

Collection Javadoc questions
Description
The collection contains all the Javadoc related questions from Stack Overflow.
Auto-Filter
All Questions with Javadoc in Title
Export
Attributes *
Id ✖ Title ✖ Body ✖ Tags ✖ has_code ✖
Export format *
csv

Fig. 4. Dataset Preparation Interface of Makar

- **Processing**. The user can preprocess the data in Makar through *transformation steps*. A *transformation step* is a single operation that is applied to all records in a collection. Currently, the tool supports operations such as *text cleaning*, *natural language processing*, *data restructuring*, and *arithmetic and counting*.
  – In *text cleaning*, the user can strip all HTML tags, or selected HTML tags, or replace records with custom values *e.g.*, remove HTML tags from questions in SO.
  – In *natural language processing*, the user can apply word stemming,[13] remove all stop words,[14] or remove all punctuation.
  – In *data restructuring*, the user can merge records having same value, create new records, remove duplicates, split text on defined substring, add a static value. In addition, the user can create a new dataset with a randomized sample, which is widely performed in manual analysis studies.
  – In *arithmetic and counting*, the user can also perform simple arithmetic steps *e.g.*, counting frequent occurrences of a particular value or a word.
- **Data export**. The user can select which attributes are to be selected for the export, and then export the data in the required format as shown in Figure 4. Currently, the tool supports *CSV*, *JSON*, and .txt (file) formats. Makar also supports adding more complicated export formats via

[13]https://snowballstem.org/algorithms/porter/stemmer.html
[14]http://snowball.tartarus.org/algorithms/english/stop.txt

`ExportAdapter`. To perform LDA (Latent Dirichlet Allocation) analysis using Mallet, we added the Mallet adapter (custom export adapter).[15]

IV. Multi-source analysis using Makar

Code comments play a crucial role in program comprehension and maintenance [8]. However, their semi-structured nature and the availability of multiple commenting conventions confront developers with numerous ways to write them. Consequently, developers often post questions to learn about different conventions on various sources such as Q&A websites [2]. Analyzing these discussions can help us to identify the issues developers face with various commenting conventions [5], [9]. To identify such concerns, we formulated a research question, *What topics do developers discuss about commenting conventions?* To answer this question, we conducted an empirical study on SO, Quora, and mailing lists.

Methodology. We manually identified ten relevant tags from SO by searching *comment* and *convention* keywords on its tag page.[16] The selected tags are: *comments*, *commenting*, *code-comments*, *block-comments*, *autocommenting*, *comment-conventions*, *convention*, *conventions*, *coding-style*. Based on a heuristics-based approach proposed by Ying *et al.*, we added five more relevant tags: *documentation*, *todo*, *code-documentation*, *naming*, *readability* [9]. We used the relevant tags from SO as keywords to find relevant topics on Quora. As a result, we obtained five topics from Quora: *Code Comments*, *Source Code*, *Coding Style*, *Programming Languages*, *Comment (computer programming)]*. We mined mailing lists of five Apache projects that we selected based on the top programming languages, Line Of Code, and number of commits from the Apache statistics report.[17] Thus, we considered *Lucene* (Java), *Ambari* (JS), *OpenOffice* (C++), *Cloudstack* (Python), and *Subversion* (C). From these projects, we mined *@dev*, *@users* and *@docs* mailing lists. The resulting data from each source is shown in Table II.

To obtain the high-level overview for SO questions, we use the popular topic analysis method, LDA [4]. After manually analyzing the results with LDA hyperparameters k, α, β set to various values we chose the values $k = 14, \alpha = 5, \beta = 0.01$ for our study. We ran LDA for 1,200 sampling iterations to stabilize the results. We reported the topics retrieved from LDA and their relevancy in Table III. We also manually analyzed a statistically significant sample set of discussions from each source (reaching 95% confidence level and an error margin of 5%). Three authors classified the sampled questions into various categories (inspired by the categories of Beyer *et al.* [10]) using a closed cart sorting [11] and majority voting approach. Due to low number of posts from Quora, we manually analysed all 689 of them.

[15]http://mallet.cs.umass.edu/

[16]https://stackoverflow.com/tagsverifiedon20Nov2020

[17]https://projects.apache.org/statistics.html

TABLE II
Data extracted from various sources

Source	Fields extracted	Candidate posts	Manually analyzed
SO	id, title, body, tags, creation date, view count	11 931	373
Quora	url, title, body, topics, answers	689	689
Mailing lists	all	140 667	385

TABLE III
Topics presented by LDA

Topic	Relevant
Exceptions	✔
Documentation Generation	✔
IDE & Editors	✔
Processing Code Comments	✔
Testing	✖
Project Documentation	✔
Project Naming Conventions	✖
Naming Conventions of Code Entities	✖
Comments Writing Strategies	✔
Comment Functionality Websites	✖
Comment Framework	✖
Database	✖
Readability	✖
Comments Syntax	✔

Findings. The results from the LDA analysis in Table III showed that not all of the generated topics are relevant to code comments (marked ✖). Analyzing the top 20 ranked questions of each topic showed that various tags such as "comment" and "commenting" are ambiguous as they contain diverse questions *e.g.*, about Facebook comments, or about the frameworks used to develop the commenting feature for the websites, thus suggesting a careful human intervention is needed to select the relevant questions from the tag. On the other hand, some tags, such as "naming" and "readability", rarely contain comment-related questions, thus contributing more to irrelevant topics in Table III, and suggesting their exclusion from further comment related studies.

Our manual analysis results highlights that developers discuss comment-related questions most frequently on SO, less frequently on Quora, and rarely discuss them in mailing lists. This is the first study to investigate code comment-related discussions on multiple sources. We also found that different kinds of questions are prevalent on SO and Quora as shown in Figure 5. Developers ask questions about how to write and format comments (implementation strategies) in various tools and IDEs more on SO compared to Quora. On Quora, we observed questions about best practices and background information about comments.

Makar supported us in preparing the dataset suitable for the LDA analysis and manual analysis.

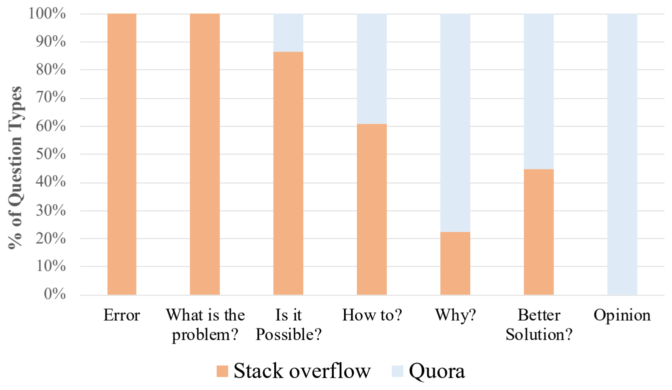
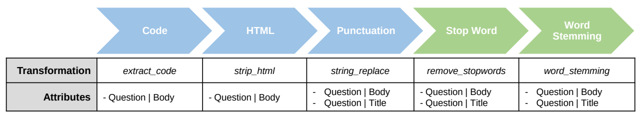


Fig. 5. Types of questions developers ask on SO and Quora

- **Data import**: We imported the SO data using the CSV import adapter, Quora data with the JSON adapter, and mailing lists with the *Apache Mailing List Archive* adapter. The CSV files of the dataset are provided in the Replication package.[18]
- **Data preprocessing**: The data from SO contains HTML, code snippets, links and natural language text. To get meaningful results from LDA analysis, the data need to be cleaned, with *text cleaning* and *language cleaning* steps. All preprocessing steps such as removing code, HTML, punctuations, and stop words,[19] and stemming words[20] are performed by Makar using its built-in transformations as shown in Figure 6. In the figure, the *Transformation* as described in section III, shows various built-in transformations of Makar and *Attributes* shows the list of selected fields (*e.g.*, Title, Body) from the sources. Each transformation is designed to produce a new attribute (a column) in the data records, allowing us to retrace the changes applied to the data. As it is generally uncertain in the beginning of a study which combination of preprocessing steps would lead to the best results, the flexible approach of Makar supported us in trying several scenarios efficiently.



Transformation	*extract_code*	*strip_html*	*string_replace*	*remove_stopwords*	*word_stemming*
Attributes	- Question | Body	- Question | Body	- Question | Body - Question | Title	- Question | Body - Question | Title	- Question | Body - Question | Title

Fig. 6. Preprocessing steps with the transformation in the tool

- **Data export**: The dataset from the case study has been exported as CSV and provided in the replication package.[21]

[18]https://doi.org/10.5281/zenodo.4281467
[19]http://snowball.tartarus.org/algorithms/english/stop.txt
[20]https://snowballstem.org/algorithms/porter/stemmer.html
[21]"Data" folder in https://doi.org/10.5281/zenodo.4281467

Conclusion

In this paper, we presented Makar, a tool supporting and enabling multi-source empirical studies. The performance of the tool has been assessed through an empirical study involving 11 931 questions from SO, 140 667 mails from mailing lists, and 700 from Quora. Makar helped us to process the multi-source data in a uniform way and to investigate various combinations of features for both LDA analysis and manual analysis. Moreover, Makar provides an extensible framework to support custom requirements, so further textual analysis techniques can be integrated to perform advanced text operations.

References

[1] M. Lehman, D. Perry, J. Ramil, W. Turski, and P. Wernick, "Metrics and laws of software evolution–the nineties view," in *Proceedings IEEE International Software Metrics Symposium (METRICS'97)*. Los Alamitos CA: IEEE Computer Society Press, 1997, pp. 20–32.

[2] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in Stack Overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014. [Online]. Available: https://doi.org/10.1007/s10664-012-9231-y

[3] G. Bavota, "Mining unstructured data in software repositories: Current and future trends," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 5. IEEE, 2016, pp. 1–12.

[4] T.-H. Chen, S. W. Thomas, and A. E. Hassan, "A survey on the use of topic models when mining software repositories," *Empirical Softw. Engg.*, vol. 21, no. 5, pp. 1843–1919, oct 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9402-8

[5] S. Amann, S. Beyer, K. Kevic, and H. Gall, *Software Mining Studies: Goals, Approaches, Artifacts, and Replicability*. Springer International Publishing, 2015, pp. 121–158. [Online]. Available: https://doi.org/10.1007/978-3-319-28406-4_5

[6] J. M. González-Barahona and G. Robles, "On the reproducibility of empirical software engineering studies based on data retrieved from development repositories," *Empirical Software Engineering*, vol. 17, no. 1, pp. 75–89, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10664-011-9181-9

[7] J. Richner, "Software developers' information needs," University of Bern, Bachelor's thesis, Feb. 2019. [Online]. Available: http://scg.unibe.ch/archive/projects/Rich19a.pdf

[8] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira, "A study of the documentation essential to software maintenance," in *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information*, ser. SIGDOC '05. New York, NY, USA: ACM, 2005, pp. 68–75.

[9] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of Stack Overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016. [Online]. Available: https://doi.org/10.1007/s11390-016-1672-0

[10] S. Beyer and M. Pinzger, "A manual categorization of Android app development issues on Stack Overflow," in *Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution*, ser. ICSME '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 531–535. [Online]. Available: http://dx.doi.org/10.1109/ICSME.2014.88

[11] E. Miller, "Universal Methods of Design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions," 2012.