6-2018

LibraryGuru: API recommendation for Android developers

Weizhao YUAN

Hoang H. NGUYEN
Singapore Management University, hhnguyen@smu.edu.sg

Lingxiao JIANG
Singapore Management University, lxjiang@smu.edu.sg

Yuting CHEN
Shanghai Jiaotong University

DOI: https://doi.org/10.1145/3183440.3195011

Poster: LibraryGuru: API Recommendation for Android Developers

Weizhao Yuan*+, Hoang H. Nguyen*, Lingxiao Jiang*, Yuting Chen+
*School of Information Systems, Singapore Management University, Singapore
+Department of Computer Science and Engineering, Shanghai Jiao Tong University, P.R. China
weizhaoy@163.com,hhnguyen@smu.edu.sg,lxjiang@smu.edu.sg,chenyt@cs.sjtu.edu.cn

ABSTRACT

Developing modern mobile applications often require the uses of many libraries specific for the mobile platform, which can be overwhelmingly too many for application developers to find what are needed for a functionality and where and how to use them properly. This paper presents a tool, named LibraryGuru, to recommend suitable Android APIs for given functionality descriptions. It not only recommends functional APIs that can be invoked for implementing the functionality, but also recommends event callback APIs that are inherent in the Android framework and need to be overridden in the application. LibraryGuru internally builds correlation databases among various functionality descriptions and Android APIs. These correlations are extracted from Android development tutorials and SDK documents with domain-specific code parsing and natural language processing techniques adapted for functional APIs and event callback APIs separately, and are matched against functionality queries to recommend relevant APIs for developers.

LibraryGuru is publicly accessible at http://libraryguru.info, and a demo video is available at https://youtu.be/f7MtjliUM-4.

1 INTRODUCTION

New software developers often need helps in getting familiar with the library APIs provided for the environment to implement required functionalities. In an event-driven programming framework, such as the Android framework, they need to know not only the APIs that can be invoked for implementing a functionality, which we call *functional APIs*, but also the event callback APIs that are defined in the framework and can be overridden to customize application behaviours, which we simply call *callbacks*. Functional APIs tell developers *what* to do for implementing a functionality, while callbacks tell developers *where* to place the implementation code. Both kinds of APIs need to be used appropriately *in combination* for implementing a functionality in the Android framework, where control flows or call relations among the APIs may not be explicit.

Based on an empirical analysis of about 1500 questions from StackOverflow.com that contain the "Android" tag, we noted that about 35% (540) of the questions have one or more answer that contains callbacks (e.g., `onClick`, `doInBackground` etc.), even though only about 50 explicitly ask when or where to implement a functionality. We thus realized that new Android programmers seldom have enough experience to understand the event-driven programming style in Android and the callbacks related to their tasks; they tend to ask high-level questions and are unlikely to ask callbacks even though implementing their functionality requires them to override some callbacks, as implied by the ten times difference between the two set of questions above (540 vs. 50). Therefore, we think recommending the necessary callbacks can be different from recommending functional APIs and both are useful for developers.

This paper presents a new API recommendation engine for Android, named LibraryGuru, that can recommend both (1) functional APIs for implementing feature requirements and (2) the callbacks that are suitable to be overridden to contain the functional code.

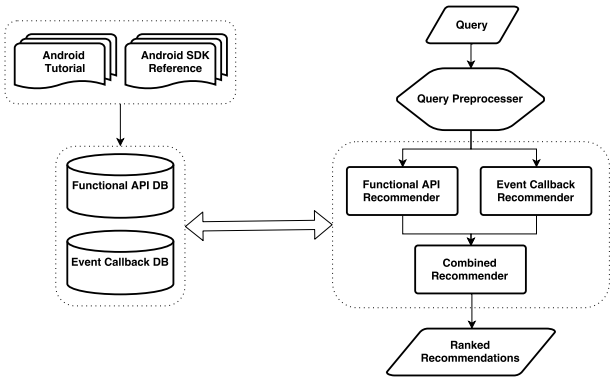


Figure 1: Workflow of LibraryGuru

2 RELATED WORK

LibraryGuru is related to numerous studies on code search and recommendation for various software development and maintenance tasks using a variety of data [3, 4, 6–8]. LibraryGuru differs from other code recommendation techniques and tools in (1) adapting the techniques for recommending callbacks that need overriding, and (2) using data from Android official tutorials and SDK documents, which may be arguably the most reliable set of reference materials for Android APIs, for building recommendation databases.

3 TOOL DESIGN & IMPLEMENTATION

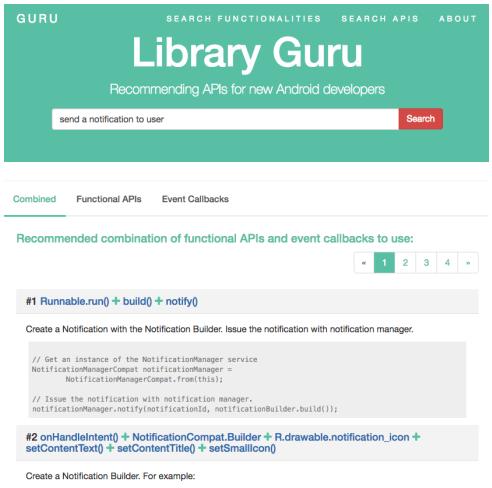
Overview. The overall workflow of LiraryGuru (Figure 1) mainly consists of two phases: The first phase constructs API description and correlation databases by extracting information from the official Android developers' tutorials[1] and SDK references[2]. In the second phase, LibraryGuru recommends functional APIs and callbacks for a given query. It takes as input a textual query which can be a sentence or paragraph describing the needs for a certain functionality, and returns a list of relevant functional APIs and callbacks accordingly.

Description & Correlation Extraction. The Android tutorial and SDK web pages contain sample code and textual descriptions.

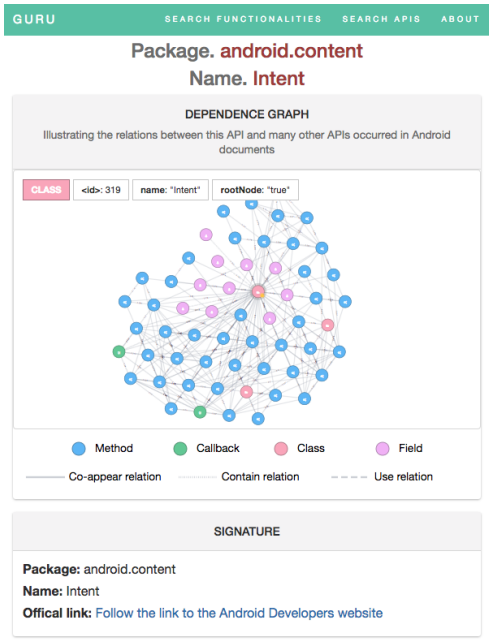
[1]https://developer.android.com/training/index.html
[2]https://developer.android.com/reference/packages.html

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195011



(a) Sample Query Page and Recommendation Results



(b) Sample Detail Page for an Identified API

Figure 2: LibrarryGuru User Interfaces

With customized text pre-processing and code parsing, we heuristically utilize the spatial distances among the APIs and descriptions to establish correlations among them, and the heuristics for finding descriptions for functional APIs and callbacks are different as they often appear in different parts in code and texts. Finally, the correlation databases can contain relevant information for each API and provide potential matches for given queries.

Similarity & API Recommendation. We adapt the *vector space model* [5] to represent API descriptions and queries. With heuristic text processing, we convert each description and query into a variant of *Term Frequency-Inverse Document Frequency* (TF-IDF) vectors, and use *WordNet* [2] to find synonyms for major words selected according to their part of speech in a query and produce multiple sub-queries using different synonyms. Similarities between the vectors of the queries and the API descriptions are then measured by *Cosine Similarity* to get nearest matches for the queries and re-rank them according to their similarity scores.

All the processing are implemented in Java and Python on top of Eclipse JDT and *NLTK* [1].

4 USAGE VALIDATION & DISCUSSION

Figure 2 illustrates the user interfaces of LibraryGuru. Figure 2(a) indicates the recommended results for a query "send a notification to user". The top 1 result in the "combined" tab indeed shows how a developer could implement the functionality with sample code: it can be done by overriding the `run` method in a `Runnable` thread that builds a notification and notifies a receiver by invoking the `build` and `notify` APIs. A user can also click the links on the API names for more details. Among the signature and sample usages, Figure 2(b) shows the correlations contained in our backend databases for the `Intent` class, in the form of a dependence graph.

Although our techniques can be applicable for other event-driven programming frameworks, we leave a larger scale evaluation for future. Many other code search and recommendation techniques may be adapted for callbacks too. However, the differences may become more apparent when we aim to recommend *how* to use the recommended APIs, in addition to *what* to use. It remains an interesting study for us to compare the performance of many different search engines adapted for callbacks, and incorporate the advantages of various techniques together to improve recommendation results and potentially generate API usage samples too.

5 CONCLUSION & FUTURE WORK

This paper shows the need and a tool named LibraryGuru for recommending both functional APIs and callbacks for Android application development based on Android tutorials and SDK documents.

In the near future, we want to improve LibraryGuru by enhancing description generation and query matching with more accurate domain-specific synonyms, correlations, and concept maps, and more semantic-aware code and language parsing and analysis of more available code samples.

ACKNOWLEDGEMENTS

This work was supported by the Singapore MOE AcRF Tier 1 grant and the China NSFC grant No. 61572312. We greatly appreciate the supports and useful feedback from anonymous reviewers.

REFERENCES

[1] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *ETMTNLP*. 63–70.
[2] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41.
[3] Mohammad Masudur Rahman, Chanchal Kumar Roy, and David Lo. 2016. RACK: Automatic API Recommendation Using Crowdsourced Knowledge. In *SANER*, Vol. 1. 349–359.
[4] Martin P. Robillard, Walid Maalej, Robert J Walker, and Thomas Zimmermann (Eds.). 2014. *Recommendation Systems in Software Engineering* (1 ed.). Springer-Verlag Berlin Heidelberg. https://doi.org/10.1007/978-3-642-45135-5
[5] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620.
[6] Ferdian Thung, Shaowei Wang, David Lo, and Julia L. Lawall. 2013. Automatic recommendation of API methods from feature requests. In *ASE*. 290–300.
[7] J. Wu, L. Shen, W. Guo, and W. Zhao. 2016. How Is Code Recommendation Applied in Android Development: A Qualitative Review. In *SATE*. 30–35.
[8] Junwei Wu, Liwei Shen, Wunan Guo, and Wenyun Zhao. 2017. Code recommendation for android development: how does it work and what can be improved? *Science China Information Sciences* 60, 9 (2017).