WikifyDocs: Addressing Ambiguity through Definitions in API Documentation

Andrea Chamorro
Department of Computer Science
University of Colorado Boulder
Boulder, CO, USA
anch9699@colorado.edu

Laura Moreno
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
lmorenoc@colostate.edu

***Abstract*— Application Programming Interface (API) documentation is meant to help developers understand and use the functionalities of an API. It is often assumed that developers who refer to API documentation have an inherent degree of knowledge of its terminology. However, this approach overlooks the various expertise levels that developers consulting documentation might have. Our goal is to bridge the gap between knowledge profiles of API developers and API users, by identifying potentially ambiguous vocabulary and providing contextual definitions. The presented tool is part of an overall effort to automatically improve documentation in the software engineering community.**

Keywords—Accessibility, developer documentation, developer expertise, wikification

I. Introduction

When learning a new API, developers sometimes use a concept-oriented approach [1], where any attempt to implement code comes only after a developer has thoroughly understood the concepts related to the API. However, current API documentation tends to focus on other types of knowledge (e.g., functionality or structure) rather than on concepts [2]. This might have an impact in the amount of time spent by developers in program comprehension, which is already considerable [3].

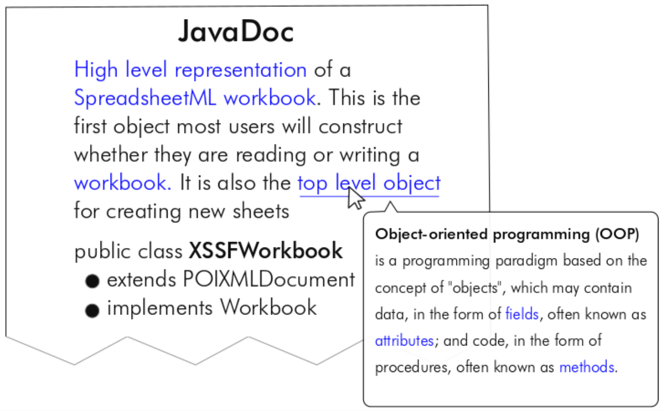
Including conceptual information in API documentation is far from trivial. As a case in point, let us examine Javadoc, a widely used tool for creating API documentation in Java programming. This tool was mainly designed to aid in the documentation of the Java Platform, SE and JDK API, whose audience is assumed to have a certain degree of programming knowledge. Therefore, its focus is on the specification of boundary conditions, argument ranges and corner cases [4]. As a result, the current Javadoc syntax does not support conceptual information (e.g., programming definitions) within the documentation and developers must turn to other documents for programming guidance. Ideally, conceptual information should be generated and non-intrusively integrated into the API documentation. We propose leveraging the Wikipedia API to provide relevant and contextually-accurate definitions for ambiguous terminology encountered in documentation. This process is also known as *text wikification* [5].

Existing tools to perform wikification, such as DBpedia Spotlight [6], have an emphasis on defining ambiguous *name entities* in general-audience documents (e.g., defining if Columbia refers to the province or the university in a news article). Our novel approach leverages past research [5], [6] and applies it to the software domain, emphasizing poly-word recognition instead of mono-word recognition.

II. WikifyDocs

A. Overview

WikifyDocs is a tool for augmenting existing Javadoc documentation with concept definitions and contextual information. In a nutshell, WikifyDocs works in the following way. Given as input a Java file (or files) from an API source code, it parses the file to extract the documentation comments of the class (or classes) in it. Then, WikifyDocs identifies ambiguous word grouplets (henceforth phrases) in the comments, which are used to query the Wikipedia API. Finally, it processes and extracts information from the Wikipedia API and outputs the documentation page in HTML format, where the definitions are only displayed when one hovers over hyperlinked phrases (see Fig. 1).



JavaDoc

High level representation of a SpreadsheetML workbook. This is the first object most users will construct whether they are reading or writing a workbook. It is also the top level object for creating new sheets

public class **XSSFWorkbook**
- extends POIXMLDocument
- implements Workbook

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.

Fig. 1. Sample output of WikifyDocs

B. Identifying Key Phrases

We denote a *key phrase* as an ambiguous phrase in the documentation that needs to be defined. In this instance of WikifyDocs, we equate ambiguity of a phrase to its complexity. We measure the complexity of a phrase by applying weights that depend on a variety of factors including (i) *phrase length;* (e.g., seven or more characters is considered complex); (ii) whether or not it has *highly contextual definitions*, as determined by the number of surface forms it takes in Wikipedia; and (iii) whether it is the *object of an action verb* (verbs in documentation tend to indicate terminology of high importance will proceed). Phrases

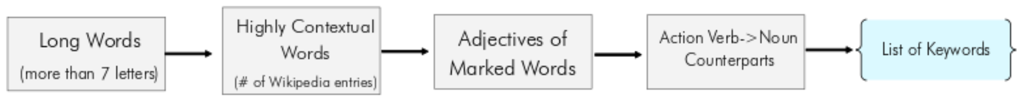


Fig. 2. Key phrase identification process in WikifyDocs

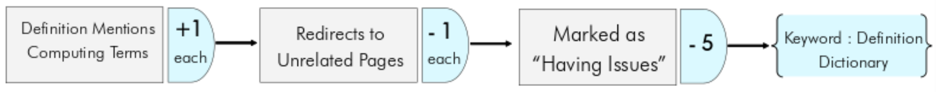

Definition Mentions Computing Terms
+1 each
Redirects to Unrelated Pages
- 1 each
Marked as "Having Issues"
- 5
Keyword : Definition Dictionary

Fig. 3. Key phrase definition process in WikifyDocs

whose complexity is higher than a threshold are marked for definition. We never define prepositions, conjunctions or common English words. The key phrase identification process is depicted in Fig. 2.

C. *Defining Key Phrases*

When looking for the definition of a phrase, WikifyDoc prioritizes definitions in the computing context and presents the one that mentions the least irrelevant entities. This is determined by the number of *page redirects* to other Wikipedia entries. It also prioritizes the entries that have the least complex definitions. Our tool takes into account if pages are tagged as having *issues* and deducts weight from them. Definition are made contextually relevant by positively weighing entries that *mention computing terms*. The tool extracts and presents the abstract of the most heavily weighted definition. The key phrase definition process is depicted in Fig. 3.

III. Discussion and Future Work

Providing definitions and contextualizing information for object-oriented classes in-time in documentation could improve the value of current documentation. We expect to bring to light three key concepts we believe are true:

1. Most information known by the source code developer can be traced back to a set of information stimuli (such as terminology) that could again be presented in a logical order to the end user developer to facilitate understanding.
2. Presenting information to the user in hover boxes, on demand, adds value to information.
3. Novice programmers can benefit from documentation that provides timely in-text definitions of basic programming concepts.

Evaluating how the tool affects document usability for this audience is our most imminent step. The implications of this research could extend into education and might help students learn programming terms more efficiently.

Our preliminary evaluation will focus on programmers with less than three years of programming experience. We hypothesize that the definition and contextually-relevant information service provided by WikifyDocs will be most beneficial, although not exclusively, for these audiences. For Chamorro (the first author), who has less than one year of programming experience, definitions were essential in the conceptual understanding of Apache POI. We expect this trend to continue for computer science students with varying expertise.

A problem cited in the past [4] with providing definitions within developer documentation is that it might detract focus and become cluttering for developers with higher levels of expertise. We address this issue in WikifyDocs by presenting definitions only when an action is performed, and by leaving the original documentation relatively unaltered.

As we continue the research in augmenting developer documentation, we expect to refine the definition of "ambiguity" within developer documentation. Moreover, we plan to add other sources of information to extract more accurate definitions, such as computing-specific dictionary databases and resources specific to the API domain (e.g., Microsoft Office documentation for Apache POI). This approach emphasizes that *understanding the domain concepts a class represents* translates into *increased understanding of the class, its functionality and relationships*. Lastly, we plan to build a *customized key phrase dictionary* for each user by collecting user information. Should this information be collected, it would help us define *learning patterns* in developers, and our efforts could be further individualized. We expect that this can be a step towards making documentation more accessible to many more developers.

Acknowledgment

We thank NSF REU (Research Experience for Undergraduates) and the CO-WY AMP for providing funding for this project.

References

[1] M. Meng, S. Steinhardt, and A. Schubert, "Application Programming Interface Documentation: What Do Software Developers Want?:," *Journal of Technical Writing and Comm.*, vol. 48, no. 3, pp. 295–330, Jul. 2017.
[2] W. Maalej and M. P. Robillard, "Patterns of Knowledge in API Reference Documentation," *IEEE Transactions on, Software Engineering*, vol. 39, no. 9, pp. 1264–1282, 2013.
[3] R. Minelli, A. Mocci, and M. Lanza, "I Know What You Did Last Summer - An Investigation of How Developers Spend Their Time," in Proceedings of IEEE 23rd Int'l Conf. on Program Comprehension, 2015, pp. 25–35.
[4] Oracle, "How to Write Doc Comments for the Javadoc Tool." Available: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html
[5] R. Mihalcea and A. Csomai, "Wikify!: Linking Documents to Encyclopedic Knowledge," in Proceedings of the Sixteenth ACM Conf. on Information and Knowledge Management, 2007, pp. 233–242.
[6] P. N. Mendes, M. Jakob, A. Garc ıa-Silva, and C. Bizer, "DBpedia Spotlight: Shedding Light on the Web of Documents," in Proceedings of the 7th Int'l Conf. on Semantic Systems, 2011, pp. 1–8.