# On the Similarity of Software Development Documentation

Mathias Ellmann
University of Hamburg
Hamburg, Germany
ellmann@informatik.uni-hamburg.de

## ABSTRACT

Software developers spent 20% of their time on information seeking on Stack Overflow, YouTube or an API reference documentation. Software developers can search within Stack Overflow for duplicates or similar posts. They can also take a look on software development documentations that have similar and additional information included as a Stack Overflow post or a development screencast in order to get new inspirations on how to solve their current development problem. The linkage of same and different types of software development documentation might safe time to evolve new software solutions and might increase the productivity of the developer's work day. In this paper we will discuss our approach to get a broader understanding of different similarity types (exact, similar and maybe) within and between software documentation as well as an understanding of how different software documentations can be extended.

## CCS CONCEPTS

• **Information systems → Content analysis and feature selection**; **Similarity measures**; **Document filtering**; **Clustering and classification**; *Deduplication*; Data cleaning; Clustering; • **Software and its engineering → Reusability**; **Documentation**;

## KEYWORDS

Software Development Documentation, Similarity Types, Software Analytics

**ACM Reference Format:**
Mathias Ellmann. 2017. On the Similarity of Software Development Documentation. In *Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17),* 4 pages.
https://doi.org/10.1145/3106237.3119875

## 1 INTRODUCTION

Nowadays developers do not only use documentations such as wikis, project work logs [26] or API reference documentations [27] to develop software. Developers also use documentations and their embedded media files that are created and maintained by the software community active on websites as Stack Overflow, [1] Eclipse Bugzilla [2] or YouTube [3] [4, 8, 29, 38]. Software developers create over 3,000 [4] software documents [3, 11] of a software documentation as development posts every day. This might lead to software development documents that can be similar to each other by their characteristics or knowledge included. The software documents might be synthetically or even semantically similar to each other as code clones [21] are and can be distinguished into different similarity types such as duplicate, similar or maybe similar.

The software community is often unsure how and when to declare a software document as similar. For example the Stack Overflow community [1, 2] discusses when a Q&A can be considered as a duplicate and when it's a usual post. This uncertainty might lead to a point where similar software documents will be deleted which can be very useful for software developers [8]. To understand the similarity of software documents there is a need of empirical studies such as a content analysis, interviews or other empirical methods [22, 30].

Software developers spent 20% of their time on searching for information [20, 36] on websites as Stack Overflow or others. They take a look on similar software documentations that include similar and additional information as an API reference document and/or Stack Overflow post to get new inspirations on how to solve their current development problem. The linkage of different types of documentations might safe time to evolve new software solutions and might increase the productivity of the developer's work day [25].

The reminder of this paper is structured as follows. In Section 2 we will describe the research problem and the challenges we will face during the thesis. In Section 3 we will describe our research method and procedure to solve the research problem. In Section 4 we will report on the current results. Section 5 concludes the paper.

## 2 RESEARCH PROBLEM AND CHALLENGES

The existence of similar software documents within a software documentation is a commonly known problem in the domain of software engineering [4, 8]. In the meantime, the software community is often unsure how to handle and evaluate similar software documents [2, 7] in Stack Overflow (SO) because of missing evaluation criteria and the complexity of understanding the characteristics of similar software development documents. Also other researchers as Roy et al. [34] try to find similar software artifacts as code clones distinguishing them into different types of similarities.


*ESEC/FSE'17, September 4–8, 2017, Paderborn, Germany*

ACM ISBN 978-1-4503-5105-8/17/09…$15.00
https://doi.org/10.1145/3106237.3119875

[1] https://stackoverflow.com
[2] https://bugs.eclipse.org/bugs/
[3] https://www.youtube.com
[4] https://api.stackexchange.com/2.2/info?site=stackoverflow


**Figure 1: Similarity types.**

Developers use different information resources to develop software [23]. Every software document of a software documentation contains different information entities as a question, an instruction or even a sequence of statements. Software documents of software documentations can contain different information types as text, code, images, audio or videos that can be useful to perform a certain development task which can actually extend each other in order to reduce their time seeking for information [20, 36].

Software development documentation is created to support development activities in a certain development context [17, 24, 28]. We are studying software development documentations that were created and/or maintained by software communities active in Stack Overflow, in Eclipse Bugzilla and on YouTube. We will study and report on the similarity as well as on the dissimilarity of development posts, development tasks and development screencasts that contain different information entities (see Table 1) and report on our research approach to study three types of similarities: duplicate, similar and maybe similar. Finally, we will inquire how different software development documentations can be assigned to the three types of similarities and enriched by each other.

In this thesis we will distinguish between three different types of similarities (see Figure 1) in three software development documentations. In the first type, we evaluate when software development documents are exactly similar to each other. We call those software documents duplicates. A duplicate exists if the pre-dominant characteristics of software development documents are the same. This can also be the case if the question is the same but the answers and steps to perform them slightly differ. When documents can be enriched by each other we call them similar software development documents. A similar software document can enrich another software document maybe because it covers the same topic. For instance a post in Stack Overflow as "how to change the UI color" might be extended by another post which is called "how to add UI components" that does not discuss the same problem. A maybe similar software document can indirectly be associated with another document e.g. when similar actions were taken to perform a development task in different programming languages.

# 3 RESEARCH METHOD

## 3.1 Research procedure

In our research we conduct a content analysis [22] to reach a better understanding within and between similar software documentation based on content as text, links or multimedia files. We download posts from Stack Overflow, [5] development tasks from Eclipse Bugzilla [6] and development screencasts from YouTube [7], pre-process them and study the characteristics as well as their syntactical and semantical similarity [4, 13, 18, 31]. To get a better understanding how developers evaluate similar development tasks we conducted interviews with computer science students [25].

**Table 1: Software development documentations and their information entities.**

| Software Documentation | Information Entities |
|---|---|
| Posts | Question, Description, Tags, Answer |
| Tasks | Summary, Description, Context, Meta-data as product name |
| Screencasts | Summary, Description, Transcript, Video Content |

## 3.2 Research Data

A software development documentation is a set of documents in which developers provide information for software development. The software development document can contain different information entities such as a question, a summary of a task or a video (see Table 1). The information entities can contain different types of information as a text, a code or links to other software documents. Software development documentations can enrich each other by their different information types as a task which has a specific task description and can be enriched by a video that provides a step-by-step instruction [29] on how to perform the development task.

We exclude code-clones [34] in our analysis because we want to understand the different types of similarities of similar software development documentations. Therefore we only consider text, links and/or the developers' expertise [5, 6]. We will also not focus API reference documentation [27] because the main information in this software documentation is code. We also exclude Tutorials from our analysis because we only focus on informations which are not spread over pages [37].

## 3.3 Characteristic Analysis

In the characteristic analysis we extract the information entities of a software development documentation and study the different types of information such as text or the context information [19] in software development tasks. In the analysis we consider the text and calculate different metrics as e.g. the number of characters. We also extract the links as well as topics which were mentioned trying to assign them to different types of similarity. Along with this we also analyze the community interaction in similar software documents. For example, we consider the number of views and the reputation of the developers taking into consideration the activity of the community (answering time, deletion time etc.). In this analysis we will show what characteristics are pre-dominant in order to declare software development documents as exact, similar or maybe similar.

[5] https://archive.org/details/stackexchange

[6] https://bugs.eclipse.org/bugs/

[7] https://www.youtube.com/watch?v=aChK4W406vg

### 3.4 Syntactical Analysis

In the syntactical analysis we study the syntactical similarity within software development documentations. To evaluate the syntactical similarity we use the Jaccard [18] algorithm and report on the syntactical differences as word differences. Based on the syntactical analysis we can report on syntactical factors when software documents can be considered as exact, similar and maybe similar software development documents. We also consider using Wordnet [8] which helps to find synonyms of words to understand syntactical similarities by using their synonyms.

### 3.5 Semantical Analysis

In the semantical analysis between software development documents we study how information entities are semantically similar to each other. The semantic comparison between documents is only possible by using already existing software documents. The information entities of a software document are compared with the information entities of another document. To make a semantical comparison between software development documents we use the LSI algorithm [15]. There are also other algorithms as word-to-vect [9] using neural networks to find similar concepts.

### 3.6 Classification & Prediction Analysis

After having studied the characteristics, syntactical and semantical similarity we try to automatically classify and predict software development documents for the three types of similarities. For the classification and prediction we use probability methods as decision trees [4, 14, 39] and/or learning algorithms as a Naïve Bayes algorithm [9]. To evaluate the classification and prediction we use usual recommendation metrics as precision and recall [33].

### 3.7 Extension Analysis

We will study how software development documentations can extend each other. Every software development document contains different pieces of information entities as e.g. a question, a task summary or a video content. For example, we will show how a development task can be linked with an answer or a question that might be needed to perform the development task. We therefore use similarity measures as the Jaccard or LSI to find a lexical connection between both.

## 4 PRELIMINARY RESULTS

In the further proceedings we report on findings of analyzing exact and similar posts in Stack Overflow. Then we will show how computer science students evaluate similar software development tasks in general [25]. Finally, we will report on which topics can be found in general for software development screencasts [16].

### 4.1 Posts in Stack Overflow

In total, the dump - a MangoDB database - included 93.8 GB of with 7,214,697 posts posted between July 2008 and May 2014. Our data set includes 69,563 **exact** post pairs (48,480 originals and 64,562 duplicates) and 43,756 **similar** post pairs (32,021 originals and 41,290 duplicates) that were defined by the community. We also build a database with 72,399 **unique** posts that are not included in the data set and reported on the differences compared to similar software development documents. Our findings are the followings in respect to the characteristics of exact and similar software posts:

- Exact or similar posts receive a higher interest and get answered faster than usual posts independently from their types.
- Exact or similar posts include rather general background knowledge while unique posts include specific knowledge.
- Developers who post an exact or similar post have a lower reputation score and might be less experienced with Stack Overflow or in the topic of interest.
- The originals of an exact post seem to get more answers, longer answer text, and more links than the originals of similar duplicates.
- Topics in exact and similar posts are different. Exact posts discuss basic operations whereas similar posts discuss more complex, open questions.

### 4.2 Development Tasks

In this study we show how developers (computer science students) identify similar tasks as similar and evaluate how they order them based on their similarity (see also Maalej et al. [25]). To evaluate the interpretation of the similar tasks we handed out a questionnaire to 127 students. We had a 84 % response rate. More than half of the ranks exactly match with the expected positions. We could summarize the findings as follows.

- Functionality & architecture indicate task similarity. Students often compared tasks on a conceptual level, relating the design, architecture, or workflow of the desired solution.
- Different technologies let tasks appear dissimilar. Participants claimed that if the technologies (e.g., framework or programming language) involved in the tasks are different, both tasks were considered as completely dissimilar.
- Participants think about tools instead of artifacts. Participants compared tasks based on the tools that were used, but rarely referred to concrete artifacts created, modified, or managed with those tools.
- Participants think about artifact types instead of artifact instances. Instead of explaining differences between tasks based on concrete artifacts (e.g., index.html, main.css, or Message.java), 18 participants rather see the differences based on the artifact types (HTML, CSS, or Java documents).

### 4.3 Development Screencasts on YouTube

We focused on development tasks performed using the Java programming language. In particular, we searched for "how-to" development screencasts[10] on YouTube using the search string "Java + *How to*". Our data set includes 431 Java development screencasts. We used the Python toolkit pyLDAvis [12, 32, 35] to identify the topics. Using this toolkit, it is possible to visualize different software development topics and to cluster them according to a varying number of LDA topics [10, 35]. Table 2 shows the overall summary of the

[8] https://wordnet.princeton.edu
[9] https://github.com/dav/word2vec
[10] *How-tos* are also among the most requested on Stack Overflow [38]

topics. As we can see here, there are two similar topics discussed in screencasts as database operations in Java and database operations in Android. This might indicate that they might be similar or at least maybe similar software development documents in YouTube.

**Table 2: Topics of Java screencasts.**

| Topic label | Most relevant terms |
|---|---|
| 6 Topics of development screencasts (analysis of the titles) | |
| Database operation with Java | netbean, database, create, mysql |
| Database operation with Android | class, table, key, android |
| System set-up | run, make, Window, JDK |
| Plug-in development | connect, jframe, constructor, jbutton |
| Game development | game, develop, object, implement |
| Testing | selenium, use, program, file, write, learn |

## 5 CONCLUSION

In this paper we have described our research method of studying three types of similarities in three types of software development documentations. We could show that there is a difference between exact and duplicate posts e.g. by their linked documents or topics discussed. We could also show that development tasks appear similar because of the used software architecture or the used technology. We could also find indicators that development screencasts might be similar at least in some part because of similar tasks performed. Further work is needed to understand similarities in software development documentations as well as how to classify, predict and extend them.

## REFERENCES


[1] 2007. Duplicate Bugs. (2007). https://blogs.msdn.microsoft.com/alanpa/2007/08/01/duplicate-bugs/

[2] 2016. Duplicate Bugs. (2016). https://meta.stackexchange.com/questions/10841/how-should-duplicate-questions-be-handled

[3] 2017. Definition of an artefact. (2017). https://en.oxforddictionaries.com/definition/artefact

[4] Muhammad Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K Roy, and Kevin A Schneider. 2016. Mining duplicate questions in stack overflow. In *Proceedings of the 13th International Conference on Mining Software Repositories*. ACM, 402–412.

[5] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. 2013. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing* 17, 2 (2013), 76–81.

[6] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. 2012. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 850–858.

[7] Jeff Atwood. 2009. Handling Duplicate Questions. (2009). http://blog.stackoverflow.com/2009/04/handling-duplicate-questions/

[8] Nicolas Bettenburg, Rahul Premraj, Thomas Zimmermann, and Sunghun Kim. 2008. Duplicate bug reports considered harmfulâĂę really?. In *Software maintenance, 2008. ICSM 2008. IEEE international conference on*. IEEE, 337–345.

[9] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[10] Roger B Bradford. 2008. An empirical study of required dimensionality for large-scale latent semantic indexing applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 153–162.

[11] Bernd Bruegge and Allen H Dutoit. 2004. *Object-Oriented Software Engineering Using UML, Patterns and Java-(Required)*. Prentice Hall.

[12] Jason Chuang, Christopher D Manning, and Jeffrey Heer. 2012. Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*. ACM, 74–77.

[13] Jack G Conrad, Xi S Guo, and Cindy P Schriber. 2003. Online duplicate document detection: signature reliability in a dynamic retrieval environment. In *Proceedings of the twelfth international conference on Information and knowledge management*. ACM, 443–452.

[14] Denzil Correa and Ashish Sureka. 2013. Fit or unfit: analysis and prediction of'closed questions' on stack overflow. ACM.

[15] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41, 6 (1990), 391.

[16] Mathias Ellmann, Alexander Oeser, Davide Fucci, and Walid Maalej. 2017. Find, Understand, and Extend Development Screencasts on YouTube. In *Proceedings of the 3rd International Workshop on Software Analytics*. ACM.

[17] Thomas Fritz and Gail C Murphy. 2010. Using information fragments to answer the questions developers ask. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 175–184.

[18] Anna Huang. 2008. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*. 49–56.

[19] Mik Kersten and Gail C Murphy. 2006. Using task context to improve programmer productivity. In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, 1–11.

[20] Andrew J Ko, Robert DeLine, and Gina Venolia. 2007. Information needs in collocated software development teams. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*. IEEE, 344–353.

[21] E Kodhai, S Kanmani, A Kamatchi, R Radhika, and B Vijaya Saranya. 2010. Detection of type-1 and type-2 code clones using textual analysis and metrics. In *Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on*. IEEE, 241–243.

[22] Klaus Krippendorff. 2012. *Content analysis: An introduction to its methodology*. Sage.

[23] Timothy C Lethbridge, Janice Singer, and Andrew Forward. 2003. How software engineers use documentation: The state of the practice. *IEEE software* 20, 6 (2003), 35–39.

[24] Walid Maalej. 2009. Task-first or context-first? tool integration revisited. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 344–355.

[25] Walid Maalej, Mathias Ellmann, and Romain Robbes. 2016. Using contexts similarity to predict relationships between tasks. *Journal of Systems and Software* (2016).

[26] Walid Maalej and Hans-Jörg Happel. 2010. Can development work describe itself?. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. IEEE, 191–200.

[27] Walid Maalej and Martin P Robillard. 2013. Patterns of knowledge in API reference documentation. *IEEE Transactions on Software Engineering* 39, 9 (2013), 1264–1282.

[28] Walid Maalej, Rebecca Tiarks, Tobias Roehm, and Rainer Koschke. 2014. On the comprehension of program comprehension. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 23, 4 (2014), 31.

[29] Laura MacLeod, Margaret-Anne Storey, and Andreas Bergen. 2015. Code, camera, action: how software developers document and share program knowledge using YouTube. In *Program Comprehension (ICPC), 2015 IEEE 23rd International Conference on*. IEEE, 104–114.

[30] Tim Menzies, Laurie Williams, and Thomas Zimmermann. 2016. *Perspectives on Data Science for Software Engineering*. Morgan Kaufmann.

[31] Seung-Taek Park, David M Pennock, C Lee Giles, and Robert Krovetz. 2002. Analysis of lexical signatures for finding lost or related documents. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 11–18.

[32] pyLDAvis. 2014. Python library for interactive topic model visualization. (2014). https://github.com/bmabey/pyLDAvis

[33] Martin P Robillard, Walid Maalej, Robert J Walker, and Thomas Zimmermann. 2014. *Recommendation systems in software engineering*. Springer Science & Business.

[34] Chanchal K Roy, James R Cordy, and Rainer Koschke. 2009. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of computer programming* 74, 7 (2009), 470–495.

[35] Carson Sievert and Kenneth E Shirley. 2014. LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*. 63–70.

[36] Janice Singer, Timothy Lethbridge, Norman Vinson, and Nicolas Anquetil. 2010. An examination of software engineering work practices. In *CASCON First Decade High Impact Papers*. IBM Corp., 174–188.

[37] Rebecca Tiarks and Walid Maalej. 2014. How does a typical tutorial for mobile development look like?. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 272–281.

[38] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. 2011. How do programmers ask and answer questions on the web?: Nier track. In *Software Engineering (ICSE), 2011 33rd International Conference on*. IEEE, 804–807.

[39] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.