Find, Understand, and Extend Development Screencasts on YouTube

ABSTRACT

A software development screencast is a video that captures the screen of a developer working on a specific task while explaining its implementation details. Due to the increased popularity of software development screencasts (e.g., available on YouTube), we study how and to what extent they can be used as additional source of knowledge to answer developer's questions about, for example, the use of a specific API. We first differentiate between development and other types of screencasts using video frame analysis. By using the Cosine algorithm developers can expect 10 development screencasts in the top 20 out of 100 different YouTube videos. We then extracted popular development topics on which screencasts are reporting on YouTube: database operations, system set-up, plug-in development, game development, and testing. In addition, we found six recurring development tasks that are performed in development screencasts as object usage and operations as well as UI operations. Finally, we conducted a similarity analysis by considering only the spoken words (i.e., the screencast transcripts but not the text that might appear in a scene) to link API documents, such as the Javadoc, to the appropriate screencasts. By using Cosine similarity we identified 38 relevant documents in the top 20 out of 9455 API documents.

KEYWORDS

Development screencasts, API reference documents, Similarity analysis

1 INTRODUCTION

Software development is a knowledge intensive work [6, 8, 15, 24] in which developers spend a substantial amount of their time looking for information [8]—e.g., how to fix a bug or how to use an API. They access and share knowledge through various media and sources, including API documentation [14], Q&A sites, wikis, or tutorials [15, 29, 31]. Regardless of how rich or popular a single knowledge source might be, it barely satisfies all the information needs of a specific developer within a certain context [6, 11, 15].

Nowadays, there is a growing online trend to use videos instead of text to capture and share knowledge [10]. Video content, from movies to webinars and screencasts, accounts for more than half of the internet traffic[1]. Software developers are concerned with this trend as they are using more and more video resources in their job [17, 26]. In particular, development screencasts are getting popular among technical bloggers[2] and on general purpose video-sharing platforms such as YouTube.

A screencast is a *"digital movie in which the setting is partly or wholly a computer screen, and in which audio narration describes the on-screen action"* [31]. In particular, a development screencast is created by a developer to describe and visualize a certain development task [17]. Screencasts might be more comprehensive than text since they capture video and audio in particular when repeating human interactions [9] e.g. following the instructions of a developer.

The text transcript of the screencast audio contains searchable and indexable technical terms that can refer to other artifacts, such as an API or a tool. For example, the screencast presented in Figure 1 can be extended with an API document—as shown in Figure 2—since it contains references to classes, methods and other units.

Despite the variety of categories present, YouTube[3] does not yet offer the possibility to explicitely search for screencasts explaining how to accomplish a certain development task [13, 29].

A task in a development screencast can be assigned to an topic of software development, and it is usually performed within a specific development context [12, 17] such as an IDE, a web browser or a virtual machine. However, attaining such tasks often requires the consultation of API documents [14, 30].

In this study, we will tackle the following research questions:

(1) *RQ1*: Is it possible to distinctively identify a developer screencasts from other video types based on frame analysis?
(2) *RQ2*: Which development tasks are frequently performed in software development screencasts?
(3) *RQ3*: Can a development screencast be extended with relevant API reference documents?

In particular, in Section 2 we evaluate different algorithms (Jaccard, Cosine & LSI) and their performance in identifying development screencasts by simply considering the frames of a video. In Section 3, we use the visualization techniques introduced by Sievert et al. [23] and Chuang et al. [4] to identify the topics of software development and the recurring development tasks present in development screencasts. In Section 4, we analyse the similarity between a task performed in the screencast and the relevant API documents using the TaskNav tool [30] followed by manual checking. Section 6 discusses the results, while Section 7 concludes the paper and describes future work.

2 FRAME ANALYSIS

A development screencasts is a special type of video which cannot be directly searched on YouTube due to the lack of a pre-defined category. Nonetheless, a development screencast is characterised by the small number of scenes, length, and the specific actions (e.g., inputting text) performed by a developer [17]. Moreover, in their screencasts, developers use several tools (e.g., and IDE or code editor, a terminal, a browser) to perform a development task. In this section, we present how we used the information available on the frames of the video to distinguish a development screencast from other types of video.

[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html
[2] http://www.virtuouscode.com/a-list-of-programming-screencast-series/ https://www.rubytapas.com/new-list-programming-screencast-series/
[3] https://support.google.com/youtube/answer/4594615?hl=en

Figure 1: Example of a development screencast on YouTube. It contains a video (screencast), a title describing the software development task, and a transcript.

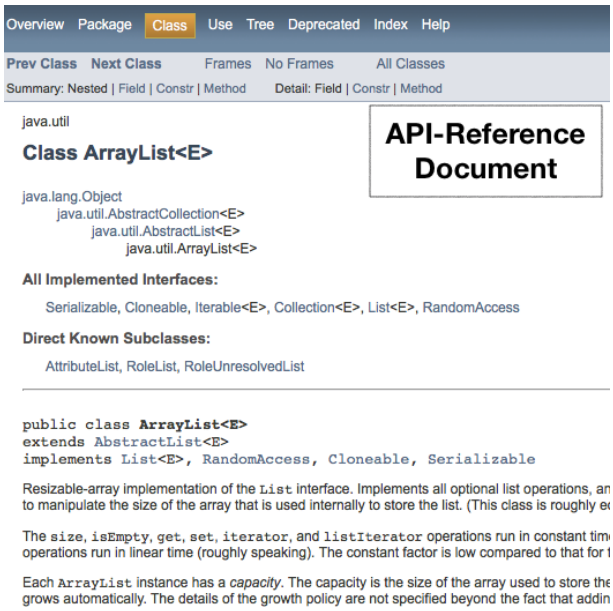

Figure 2: Example of an API reference document. It contains class and method definitions as well as the descriptions of it.

We sampled a set of frames (i.e., a rectangular raster of pixels) from different videos and compared their stability and variation. We define the similarity between two frames as $Sim(f_1, f_2)$. The frame similarity of a video is calculated by $\sum_{1}^{n} \frac{Sim(f_n, f_{n+1})}{n}$ with **n** as the number of analyzed frames. Frame f_{n+1} is the direct successor of frame f_n and $f_n \neq f_{n+1}$. We sampled a frame every 10 seconds for each video.

We randomly selected 100 YouTube videos associated to one of the following video types:

- **Development screencast (n=20)**: videos showing software development activities in several programming languages, such as PHP, Python, Java, SQL and C#. Different tools (e.g., an IDE, code editor, or simple a web browser) are used to perform a task.
- **Non-development screencasts (n=20)**: videos showing the desktop of a user solving problems unrelated to software development, including mathematical problems, game tutorials, or software utilization.
- **Non-development, non-screencast (n=20)**: videos showing how to perform a task not related to software development (e.g., learn Spanish, or how-to change a phone screen) in which a computer screen is not recorded.
- **Others (n=40)**[4]: videos in none of the above categories (e.g., a music video). This set contains 40 videos because most of the videos had a short length (2-3 minutes).

The sample for each type of video contains approxiatemly 2000 frames. Every frame contains a particular number of color information that changes in different scenes throughout the developer screencast—for example, when using an IDE, a web browser or a terminal.

The similarity between two frames was calculated using the *Jaccard* coefficient, *Cosine* similarity, and *LSI*. Each color information per pixel is considered a bag of words [32]. The *Jaccard* coefficient is used to measure the similarity between two sets of data. The cardinality of the intersection is divided by the cardinality of union of the sets [7]. The similarity value of the *Jaccard* ranges between 0 and 1. If the documents $\vec{t_a}$ and $\vec{t_b}$ contain the same set of objects it is 1. If they have no objects in common it is 0. The similarity between the two documents $\vec{t_a}$ and $\vec{t_b}$ is

$$SIM_J(\vec{t_a}, \vec{t_b}) = \frac{|\vec{t_a} \cdot \vec{t_b}|}{|\vec{t_a}|^2 + |\vec{t_b}|^2 - \vec{t_a} \cdot \vec{t_b}}$$

To measure the similarity between text documents, they can be transformed into term vectors. The *Cosine* similarity is quantified as the angle between these vectors, and ranges between -1 and 1. For text documents $\vec{t_a}$ and $\vec{t_b}$ the *Cosine* similarity is non-negative and bounded between [0,1] [7]. The *Cosine* approach is commonly used for a similarity comparison of documents [1],[5],[19]. Finally, the LSI ranging between -1 and +1—uses term frequencies for dimensionality reduction, followed by a singular value decomposition (SVD)[18]. In particular, we use the *Cosine* and *LSI* algorithms to evaluate the frequency of switches of scenes in a video. The *Jaccard* algorithm is more sensitive than *Cosine* and *LSI* as the latter two only recognize a low number of switches of scenes and moving objects (mouse, keyboard, etc.) used in the development screencasts.

We analyzed 2127 frames from 20 development screencasts. The values of *Cosine* and *LSI* are close to 1.0, indicating that in a development screencasts there is only a small number of scene switches. The Jaccard similarity has an average value of 0.768, showing that

[4]Provided from the owner of http://randomyoutube.net

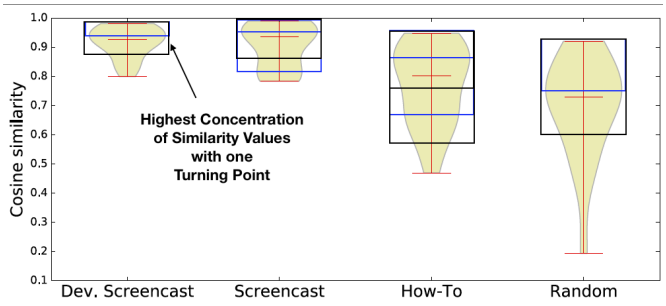

Figure 3: Frame similarity of development screencasts compared to other video types (using cosine similarity values).

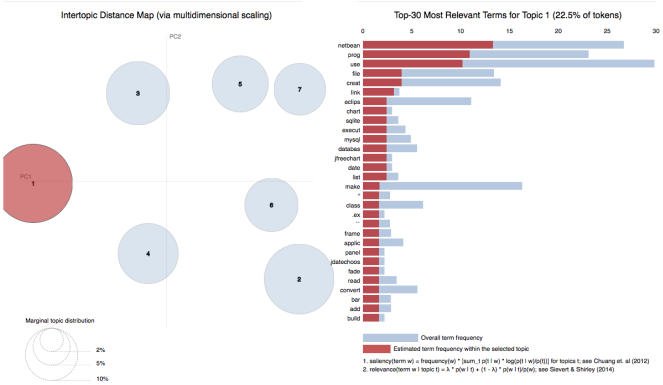

Figure 4: Topics of Java software development tasks performed in the screencasts.

small objects in the screencasts are moved. We analyzed the similarity distributions of the four sets of videos using each algorithm. The highest concentration of similar values for the development screencasts can be calculated using the Cosine algorithm (see Figure 3). Especially for the Jaccard algorithm the characteristics of the distribution varies a lot, making it difficult to clearly identify a developer screencast from other types of video. The LSI algorithm has similar distributions. However, when using the Cosine algorithm, a higher concentration, in particular for Developer Screencasts, is calculated. Thus, it is better suited to distinguish developer screencasts from other video types.

Next we identified 20 development screencast within other video types (n = 100) by using the Cosine algorithm. We calculated the frame similarity of every video of every video type and ranked the videos based on their Cosine similarity (in descending order). All developer screencasts could be predicted until the first 45 recommendations of videos because of the high concentration of the similarity values of the development screencasts in comparison to other video types. Within a list of 20 videos we could identify 55% of the development screencasts. In other words, developers can expect to correctly identify over 10 development screencast in a list of 20 different YouTube videos with the support of the Cosine algorithm. ($precision$ = 0.028, $recall$ = 0.55, and $F1$ = 0.052 [22]. To answer RQ1 we found that the Cosine algorithm is better suited to distinguish development screencast from other types of video due to its capability of better concentrating similarity values.

- Development screencasts are different then other types of videos. Development screencasts seems to be more static (have less scenes and objects which were used).
- The Cosine algorithm is the best algorithm to identify a development screencast from other video types (highest concentration of similarity values).
- All development screencasts could be identified within the first third of the retrieved items.

3 TOPICS OF SOFTWARE DEVELOPMENT SCREENCASTS

In this section, we analyze the topics of software development in which the task presented in a development screencast is performed. To this end, we analyzed the title of the screencast as well as its audio transcripts, and assigned the task performed on screen to different topics of software development, such as implementation issues or system set-up. Due to its popularity, we focused on development tasks performed using the Java programming language[5]. In particular, we searched tasks performed "how-to" development screencasts[6]. Therefore, the search string used to retrieve relevant videos from YouTube was "Java + *How to*". Our dataset includes 431 Java development screencasts; for all the videos a transcript is available. We used the Python toolkit pyLDAvis [4, 21, 23] to identify the topics of software development in which the task are performed. Using the tool it is possible to visualize different topics of software development and to cluster them by varying the number of LDA topics. We started by removing from the text all the special characters, numbers and the term "Java" which interferes with our results. We tuned the number of LDA topics until we could reach a number of non-overlapping clusters that have enough distance between each other (see Figure 4). We also modified the relevance metric λ until we found the most relevant terms for a topic of software development, as shown in Table 1.

We performed two different analysis about the topics of software development focused in software development screencasts. In the first analysis, we consider only the titles of the screencast to understand in which topics of software development the developer performs a task—e.g., system-set up. The second analysis considers the textual transcript of the development screencasts. We only consider the nouns—extracted using the NLTK library [2]. The output of this step was inaccurate as it included some verbs within this corpus; taking into consideration also verbs caused the algorithm (LDA) to overfit. For example, verbs like *make*, *use* or *create* were identified as the most relevant terms, thus hampering their abstraction to the different topics of software development.

[5] http://stackoverflow.com/tags
[6] *How-tos* are also among the most requested on Stack Overflow [28]

Table 1: Topics of and within Java screencasts.

Topic label	Most relevant terms
6 Topics of development screencasts (tasks performed in SD according to their 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
6 Topics within development screencasts (repeatable tasks performed in SD according to the transcripts)	
API usage (Object/Classes)	use, create, class, code, method, click, type
Files	file, create, call, time, program
Lists	list, move, get, create
UI operations	box, file, slider, inputs
Methods	property, get, input, statement
System operations	program, time, system, get

Database-related operations are some of the most interesting topics for which developers produce screencasts. Similarly, *MySQL* is one of the most popular topics on StackOverflow[7]. There might be a need for a system support in the IDEs to perform database tasks. Tutorials [26] as well as FAQs [27] provide a first entry to start developing a certain system. We found that such need is present in development screencasts, too. Plug-in installation is discussed in development screencasts which might extend Tutorials which provide knowledge for similar development tasks[8]. Interestingly, there is also some niche topic as game development discussed in development screencasts. Software testing is also covered in software development screencasts.

The usage and operations with a certain object, file or a class in Java is a frequently occurring topic. Method and system operations are also frequently occurring tasks during development. Especially those development screencasts might be perfect candidates to be extended with API reference documents. Interestingly, UI operations are also shown to be one of the main activities performed when comprehending software [16]. Explicitly list operations are besides usual object operations are one of the frequently occuring task which might also show the importance of this data type in comparison to similar data types as hash-maps.

- Database operations are popular development tasks performed in development screencasts.
- Testing—a conventional task in software development—can be learned as long as some help is provided to start testing using Java.
- An advanced search for a development task within software development screencasts might help to perform and follow them also independently of the overall topic of a development screencast.

4 ANALYSIS OF SIMILARITY TO API DOCUMENTATION

Our data-set contains 35 randomly chosen Java development screencasts that have high quality transcripts (no miss-spelt words etc.).

[7]http://stackoverflow.com/tags
[8]http://www.vogella.com/tutorials/EclipsePlugin/article.html

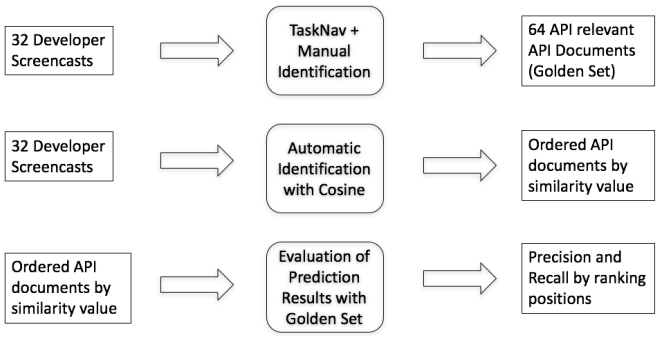

Figure 5: Research method to identify relevant API documents

Table 2: Prediction results for 65 relevant documents (pages). Search space includes 9,455 API documents.

Top	Documents Retrieved	Precision	Recall
3	18/65	0.0514	0.30
5	22/65	0.062	0.367
10	33/65	0.094	0.524
20	38/65	0.0542	0.605

We identified 1-3 relevant API reference documents for every development task that were performed in a development screencast (see research method in Figure 5). We used TaskNav to retrieve a number of relevant documents [30]. The input parameter for this tool is a phrase (i.e., the title of the screencast) that describes a certain development task. In several occasions, we could not easily find more than one relevant document because the titles of the screencast were not self-explanatory—for example, "Java How To: Dialog Boxes" or "How to make a Tic Tac Toe game in Java". A deeper look into the development screencast and its transcript was always needed. We qualitatively evaluated the recommendations of the API documents afterwards. We defined documents as relevant if they contain the same classes (e.g., ArrayList) or method signatures (e.g., boolean contains(Object o)) which were mentioned in the development screencasts as well as additional information (e.g., implement ArrayList, LinkedList), that are useful when repeating the development tasks showed in the development screencasts. We could identify 65 relevant documents from 9,455 potential candidates. For the automatic identification of the relevant development screencasts we have used the Cosine algorithm. We calculated the Cosine similarity value for each transcript of a developer screencast and each of the 9,455 Java API documents in the dataset. We received a ranked list of API documents ordered by their similarity values. We calculated the precision and recall [22] (as identified by TaskNav with manual check) within the top three, five, 10, and 20 Cosine positions, and present them in Table 2. For the best three retrieved results, we found that the transcripts frequently and clearly mention technical terms such as class and method names contained in an API documentation page. The precision values are between 5 and 10%, with the best result being yielded by the top-10 retrieved

pages. Extracting the content of the development screencast—e.g., the code showed on the screen when using an IDE—might lead to a higher precision/recall to identify a development screencast. This result is more accurate than random, which has a precision below 1%. Table 2 shows that more than 50% of the relevant documentation pages were found in the top-10 retrieved position. The percentage increases to more than 60 when the top-20 positions are considered.

- There exists a similarity threshold of relevant API documents. Above such threshold a high quantity of relevant API documents in relation to other API documents can be found.
- By comparing the audio transcript (the screencast transcripts but not the text that might appear in a scene e.g. an IDE) of a development screencast with the API documentation, we could identify 38 out of 65 relevant API documents up to the first 20 positions.
- Extracting the content of the development screencast—e.g., the code showed on the screen when using an IDE—might lead to a higher precision/recall to identify a development screencast.

5 RELATED WORK

MacLeoad et al.[17] report on the structure and content of development screencasts. They report on different types of knowledge located in screencasts. They studied the presentation of the knowledge and grouped the coding techniques to higher-level topics. In our study we focused on Java screencasts, associated them to high-level topics, conducted a frame and a similarity analysis, and discussed how screencasts can be used to enrich API reference documentation. Treude et al.[29] discuss how to link StackOverflow answers with the Java 6 SE SDK API. They use the Cosine approach to measure the similarity and then LexRank to evaluate the relevance of the API documents. We extend their work by linking screencasts with API documents and show how similar they actually are.

Ponzanelli et al.[20] developed a recommendation system to predict relevant YouTube videos[9] for Android. In addition to the audio transcripts, they used an OCR tool[10] that transfers the actual video information (e.g., slides or subtitles) to text. In their study, they focus on random YouTube videos and show the StackOverflow posts that might be relevant for such videos.

A technique for linking API documentations to code examples is introduced by Subramanian et al. [25] and Chen et. al [3]. They defined a written code as a development task. To finish the development task an API reference documentation is needed to get implementation insight of the used classes. We try to link API reference documentations to relevant development screencasts.

6 DISCUSSION AND LIMITATIONS

Based on the similarity analysis of the frames, we found that frames in a development screencast seems to be very much alike in contrast to other types of video. An identification of screencasts should, therefore, be possible by using algorithms as the Cosine similarity or LSI without knowing the actual title, tags, or the transcript of the video. Similarly, videos—such as recorded interviews or slow motion—are very static and their scenes do not change much. We acknowledge that such approach, based on frames comparison, might mistakenly find those other type of *static* videos.

The analysis of the development topics showed that development screencasts contain knowledge that is also provided in an API reference documentation. In the light of this result, API reference documents can extend a development screencast to provide additional implementation details. This is particularly interesting for those developers who avoid reading documentation [10] or prefer audio-visual learning instruments like screencasts. Leveraging our results, a simple tool—e.g., based on Cosine similarity calculation—can suggest relevant documentation pages from a large corpus like the Java SDK documentation, with a 61% recall for a list of 20 items.

We focus this preliminary study on screencasts related to a specific programming language. However, there is a wide range of other development screencast in which developers might explain things differently, or use different programming languages with their own syntax, semantics as well as specific tools. Therefore, development screencasts might differ accordingly to the tools used or when other software engineering activities and phenomena are discussed. The selection of the dataset might thus have influenced the study results.

The transcripts we obtained might miss important terms, or include misspell ones. This can impact the comparison of those transcripts with the API documentation pages, leading to poor results. Thus, we studied and manually checked only 35 screencasts and their transcripts.

Building a large dataset using the YouTube API poses some limitations. The YouTube API only returns a limited number of search results[11]. Thus, multiple searches, with different search terms, need to be performed. The persistence of retrieved data is not guaranteed due to the possible deletion of the videos included in our sample.

When performing a development task there is often the need for additional information that are gathered, for example, from Stack Overflow, YouTube or an API documentation. Combining all of them might mean to use different types of informations to perform a development task. We conclude that software development screencasts can help developers to look at recurring development tasks (e.g., within an IDE) independently from the topic of software development.

A software development screencast is a special type of video in which developers need to perform their tasks by focusing only on the most relevant tools. Development screencasts can be extended by API documents so that a developer can perform a sub-task while watching a development screencast. A similarity analysis, based only the audio transcript (e.g., excluding the screen content) can identify relevant API documents.

[9] http://codetube.inf.usi.ch/
[10] https://github.com/tesseract-ocr/tesseract/wiki
[11] http://stackoverflow.com/questions/25918405/youtube-api-v3-page-tokens

7 CONCLUSION AND FUTURE WORK

We analyzed different development screencasts on YouTube and found six main topics for the Java programming language. In particular, how-to screencasts show the steps to take in order to complete a development task. They could enrich text documentation, such as Javadocs and tutorials. Although development screencasts are not much different from other types of screencasts, we found that a high frame similarity. This distinctive characteristic can be used to detect a development screencast on YouTube. In this stage, a couple of relevant API documents are provided within a list of 10 items. A Cosine comparison between a screencast and a large API documentation corpus is only a preliminary, simple approach to help developers to match the best document.

This paper provided first insights on how to categorize and identify development screencasts, and how to enrich them with API documentation. Further work is needed to identify different types of knowledge [14, 17] located in screencasts and achieve a more fine-grained and precise mapping to the API reference documentation and the API elements within the IDE. This might require labeling every unique piece of knowledge within a screencast and using video and image features. We believe that the community needs to study which types of screencasts are useful for which developers in which situations.

REFERENCES

[1] 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.

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

[3] Cong Chen and Kang Zhang. 2014. Who asked what: Integrating crowdsourced FAQs into API documentation. In *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 456–459.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Andrew J Ko, Brad A Myers, Michael J Coblenz, and Htet Htet Aung. 2006. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on software engineering* 32, 12 (2006).

[9] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. 2008. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 1–8.

[10] 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.

[11] 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.

[12] Walid Maalej and Mathias Ellmann. 2015. On the similarity of task contexts. In *Proceedings of the Second International Workshop on Context for Software Development*. IEEE Press, 8–12.

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

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

[15] Walid Maalej, Rebecca Tiarks, Tobias Roehm, and Rainer Koschke. 2014. On the Comprehension of Program Comprehension. *ACM Transactions on Software Engineering and Methodology* 23, 4, Article 31 (Sept. 2014), 37 pages. https://doi.org/10.1145/2622669

[16] 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.

[17] Laura MacLeod, Margaret-Anne Storey, and Andreas Bergen. 2015. Code, camera, action: How software developers document and share program knowledge using YouTube. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*. IEEE Press, 104–114.

[18] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, Vol. 6. 775–780.

[19] 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.

[20] Luca Ponzanelli, Gabriele Bavota, Andrea Mocci, Massimiliano Di Penta, Rocco Oliveto, Barbara Russo, Sonia Haiduc, and Michele Lanza. 2016. CodeTube: extracting relevant fragments from software development video tutorials. In *Proceedings of the 38th International Conference on Software Engineering Companion*. ACM, 645–648.

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

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

[23] 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.

[24] Jonathan Sillito, Gail C Murphy, and Kris De Volder. 2008. Asking and answering questions during a programming change task. *IEEE Transactions on Software Engineering* 34, 4 (2008), 434–451.

[25] Siddharth Subramanian, Laura Inozemtseva, and Reid Holmes. 2014. Live API documentation. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 643–652.

[26] 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.

[27] Irmo Timmann. 2015. *An Empirical Study Towards a Quality Model for FAQs in Software Development*. Technical Report.

[28] 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.

[29] Christoph Treude and Martin P Robillard. 2016. Augmenting API documentation with insights from Stack Overflow. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, 392–403.

[30] Christoph Treude, Mathieu Sicard, Marc Klocke, and Martin Robillard. 2015. TaskNav: Task-based navigation of software documentation. In *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*, Vol. 2. IEEE, 649–652.

[31] Jon Udell. 2005. What Is Screencasting - O'Reilly Media. http://archive.oreilly.com/pub/a/oreilly/digitalmedia/2005/11/16/what-is-screencasting.html. (November 2005). (Accessed on 11/01/2016).

[32] Xiaogang Wang and Eric Grimson. 2008. Spatial latent dirichlet allocation. In *Advances in neural information processing systems*. 1577–1584.