Find, Understand, and Extend Development Screencasts on YouTube

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

Alexander Oeser
University of Hamburg, Germany
9oeser@informatik.uni-hamburg.de

Davide Fucci
University of Hamburg / HITeC, Germany
fucci@informatik.uni-hamburg.de

Walid Maalej
University of Hamburg, Germany
maalej@informatik.uni-hamburg.de

ABSTRACT

A software development screencast is a video that captures the screen of a developer working on a particular task and explaining implementation details. Due to the increased popularity of development screencasts e.g., on YouTube, we study how and to what extent they can be used as additional source of knowledge to answer developers' questions, for example about the use of a specific API. We first study the difference between development screencasts and other types of screencasts using video frame analysis. When comparing frames with the Cosine algorithm, developers can expect ten development screencasts in the top 20 out of 100 different YouTube videos. We then extracted popular development topics. These were: database operations, system set-up, plug-in development, game development, and testing. We also identified six recurring tasks performed in development screencasts, such as object usage and UI operations. Finally, we conducted a similarity analysis of the screencast transcripts and the Javadoc of the corresponding screencasts.

CCS CONCEPTS

• **Information systems** → **Clustering**; **Content analysis and feature selection**; **Similarity measures**; **Clustering and classification**; *Content ranking*; *Retrieval efficiency*; Social networks; • **General and reference** → *Empirical studies*; • **Software and its engineering** → Software libraries and repositories;

KEYWORDS

Development Screencasts, API documentation, Similarity Analytics

ACM Reference format:
Mathias Ellmann, Alexander Oeser, Davide Fucci, and Walid Maalej. 2017. Find, Understand, and Extend Development Screencasts on YouTube. In *Proceedings of 3rd International Workshop on Software Analytics, Paderborn, Germany, September 4, 2017 (SWAN'17),* 7 pages.
https://doi.org/10.1145/3121257.3121260

SWAN'17, September 4, 2017, Paderborn, Germany

ACM ISBN 978-1-4503-5157-7/17/09…$15.00
https://doi.org/10.1145/3121257.3121260

1 INTRODUCTION

Software development is a knowledge-intensive work [5, 7, 13, 22]. Developers spend a substantial amount of their time looking for information [7]—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 [12], Q&A sites [26], wikis [13], or tutorials [24]. 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 [5, 10, 13].

Nowadays, there is a growing trend to use videos instead of text to capture and share knowledge [9]. Video content, from movies to webinars and screencasts, accounts for more than half of the internet traffic[1]. Software developers are also concerned with this trend as they are using more and more video resources in their job [14, 24]. In particular, development screencasts are getting popular among technical bloggers[2] and on general purpose video-sharing platforms such as YouTube or Vimeo.

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"* [28]. In particular, a development screencast is created by a developer to describe and visualize a certain development task [14]. Screencasts are more comprehensive than plain text since they capture human interaction[8] in the form of video and audio — e.g., while a develoeper usually instruct in the voice track what to do next.

YouTube[3] does not yet offer the possibility to explicitly search for a development screencast that explains how to accomplish a specific development task [11, 26] in a certain development context [11]. Moreover, there is a lack of understanding about this type of videos and how it can be distinguished from other videos.

In a development screencast, a software developer performs a task which can be assigned to a topic and to a specific context [14], such as an IDE, a web browser or a virtual machine. The text transcript of the screencast audio track 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.

[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.

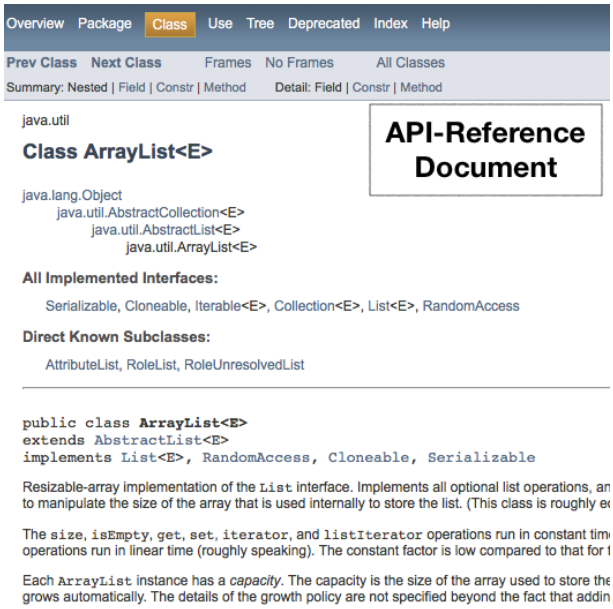

Figure 2: Example of an API reference document. It contains class and method definitions as well as the descriptions of it.

Based on the mentioned observations, we will tackle the following research questions in this paper:

(1) *RQ1*: Is it possible to distinctively identify a developer screencast from other video types?
(2) *RQ2*: Which development tasks are often performed and captured in development screencasts?
(3) *RQ3*: Can a development screencast be linked to and extended with related API reference documentation by considering only the transcripts?

In Section 2, we evaluate different algorithms (Jaccard, Cosine & LSI) and their performance in identifying development screencasts on YouTube by simply considering video frames. In Section 3, we use the topics visualization techniques introduced by Sievert et al. [21] and Chuang et al. [3] to identify frequent topics and development tasks described in screencasts. In Section 4, we analyse the similarity between a task performed in the screencast and the corresponding API documents using the TaskNav tool [27]. Section 6 discusses the results, while Section 7 concludes the paper and describes future work.

2 FRAME ANALYSIS

A development screencast 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 characterized by the small number of scenes, length, and the specific actions (e.g., coding) performed by a developer [14]. Moreover, in their screencasts, developers use several tools (e.g., an IDE, code editor, a terminal, or a browser) to perform a development task. In this section, we present how we used the information available on the video frames to distinguish a development screencast from other types of videos.

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 $\mathbf{n}$ the number of analyzed frames. Frame f_{n+1} is the direct successor of frame f_n and $f_n \neq f_{n+1}$. For each video, we sampled a frame every 10 seconds.

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

- **Development screencast (n=20)**: videos showing software development activities in several programming languages, such as PHP, Python, Java, SQL and C#.
- **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 them had a short length (2-3 minutes).

The sample contains approximately 2000 frames for each video type. Every frame contains a particular number of color information that changes in the different scenes throughout the developer

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

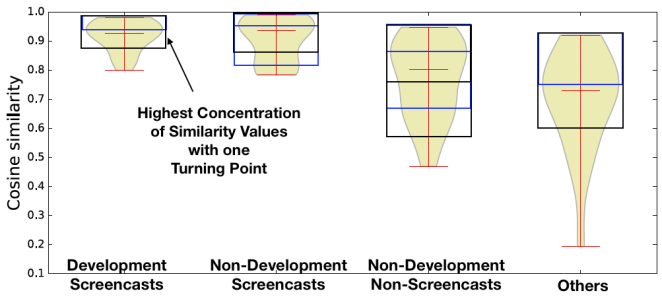

Figure 3: Frame similarity of development screencasts compared to other video types (using cosine similarity values).

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 [29]. 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 [6]. The similarity between the two documents $\vec{t_a}$ and $\vec{t_b}$ is

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

The similarity value of the *Jaccard* coefficient ranges between 0 and 1. If the documents $\vec{t_a}$ and $\vec{t_b}$ contain the same set of objects, the coefficient is one (or zero in case the documents do not have objects in common). The *Cosine* approach is commonly used for a similarity comparison of documents [1],[4],[16]. Documents are converted into term vectors to compute their *Cosine* similarity, which is the angle between these vectors and ranges between 0 and 1 [6]. Finally, the Latent Semantic Indexing (LSI) index is between -1 and +1. It uses term frequencies for dimensionality reduction, followed by a singular value decomposition (SVD) [15].

We use the *Cosine* and *LSI* algorithms to evaluate the frequency of scene switches in a video. The *Jaccard* algorithm is more sensitive than *Cosine* and *LSI* as the latter two only recognize a low number of scene switches and moving objects (mouse, keyboard, etc.) used in the development screencasts.

The analysis of 2127 frames from 20 development screencasts shows that 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 small objects 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). For the Jaccard algorithm, the characteristics of the distribution vary, making it difficult to identify a developer screencast from other types of video. The LSI algorithm has similar distributions. The Cosine algorithm shows a higher concentration in particular for Developer Screencasts then the LSI algorithm. Thus, it is better suited to distinguish developer screencasts from other video types.

We identified 20 development screencast within other video types (n = 100) by using the Cosine algorithm. We calculated the frame similarity of every video type and ranked the videos based on their Cosine similarity (in descending order). All developer screencasts are correctly predicted until the first 45 position due to the high concentration of similarity values for development screencasts with respect 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 ten development screencast in a list of 20 different YouTube videos using the conventional Cosine similarity algorithm.

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 from other types of videos. They seem to be more static—i.e., they have less scenes and objects.
- The Cosine algorithm is the best, among the studied algorithms, at identifying a development screencast from other video types (highest concentration of similarity values).

3 TOPICS OF SOFTWARE DEVELOPMENT SCREENCASTS

In this section, we analyze the software development topics of the task performed during a development screencast. To this end, we analyzed the title of the screencast as well as its audio transcripts and assigned the task performed to different software development topics, such as implementation or system set-up.

Table 1: Topics of and within 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
6 Topics within development screencasts (analysis of 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

We focused on development tasks performed using the Java programming language. In particular, we searched for "how-to" development screencasts[5] on YouTube using the search string "Java + *How to*". Our dataset includes 431 Java development screencasts; for each video a transcript is available. We used the Python toolkit

[5] *How-tos* are also among the most requested on Stack Overflow [25]

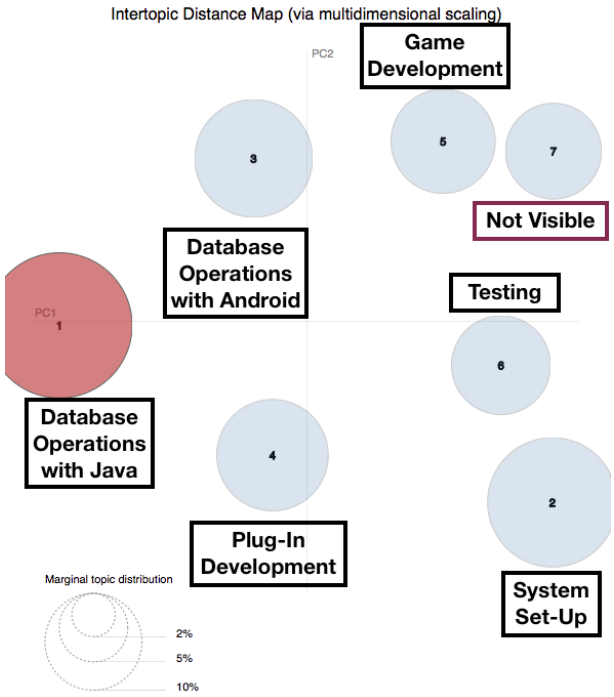

Figure 4: Topics of Java screencasts.

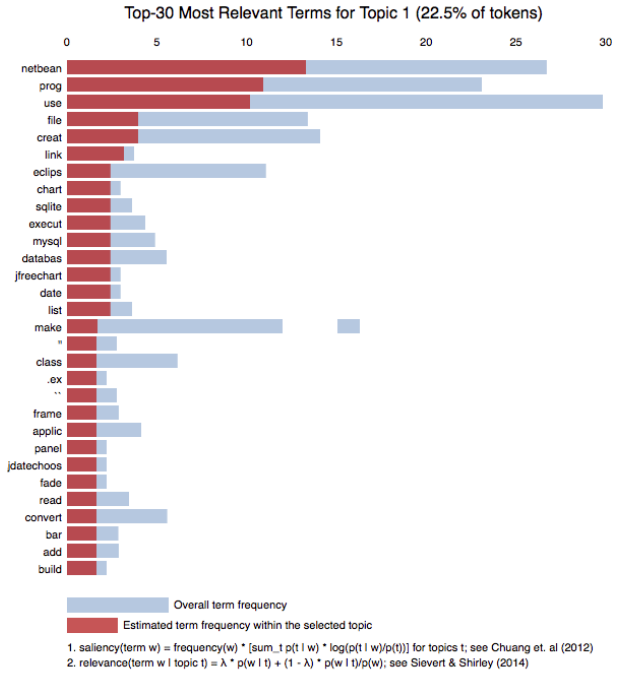

Figure 5: Distribution of terms of a screencast topic.

pyLDAvis [3, 18, 21] 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 [2, 21].

We first removed from the text all the special characters, numbers and the term "Java" which interferes with our analysis. We tuned the number of LDA topics until we reached a set 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.

We performed two different analyses of software development topics found in screencasts. In the first analysis, we considered only the titles of the screencast. The second analysis considers the textual transcript of the development screencasts.

Table 1 summarizes the topics we found in the titles and in the transcripts of the development screencasts. Figure 4 shows the clusters of all the topics within the chosen software development screencasts. We stopped searching for the best number of topics when the topic clusters did not overlap anymore or when the topics became not visible. The size of the clusters represents the importance of the topic within the overall set of topics. Figure 5 shows the distribution of terms used to derive the topic of a task.

Database-related operations are some of the most popular topics discussed in developers screencasts. Similarly, the database management system *MySQL* is one of the most popular topics discussed on StackOverflow[6]. This observation could indicate the need for a system to support answering database related questions in the IDE.

Plug-in installation is also discussed in development screencasts. This topic can extend traditional tutorials as they provide knowledge for similar development tasks[7]. We observed some niche topics, such as game development, discussed in development screencasts. Software testing, a frequent software engineering activity[23], is also covered in software development screencasts. This might reveal the need for screencasts that teach how to test software [20]. The use of a method, objects, or class in Java is a frequently occurring topic. In particular, list operations one of the most commonly occurring tasks showing the importance of this data type. Finally, UI operations are also shown to be one of the main activities.

- Database operations are popular in development screencasts.
- Testing — a conventional task in software development — is also performed in development screencasts.
- Methods and classes usage, are taught in software development screencast.

4 ANALYSIS OF SIMILARITY TO API DOCUMENTATION

Our dataset contains 35 randomly selected Java development screencasts with high-quality transcripts (e.g., no misspelled words). We identified 1-3 relevant API reference documents for every development task that was performed in a development screencast (see

[6]http://stackoverflow.com/tags

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

Figure 6). Initially, we used TaskNav: a tool for identifying relevant API documents for a specific task based on natural language queries [27]. The input parameter for this tool is the title of the screencast describing a certain development task. In several occasions, we could not find more than one relevant document because the screencast titles were not self-explanatory (for example, "Java How To: Dialog Boxes" or "How to make a Tic Tac Toe game in Java"). Therefore, we manually evaluated the recommended API documents by defining documents as *relevant* if they contain the same classes (e.g., *ArrayList*) or method signatures (e.g., *boolean contains(Object o)*) mentioned in the screencasts as well as additional useful information needed when repeating the development tasks (e.g., *implement ArrayList, LinkedList*). We could identify 65 relevant documents from 9,455 potential candidates.

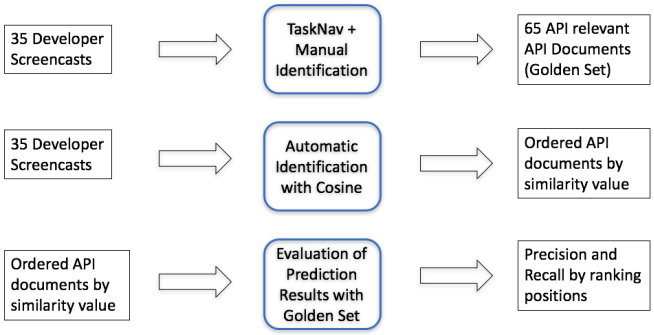

Figure 6: Method to identify relevant API documents

Table 2: Prediction results for 65 relevant documents (JavaDoc pages). The 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

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 screencast and each of the 9,455 Java API documents in the dataset which resulted in a ranked list of API documents ordered by their similarity values. For the evaluation, we calculated precision and recall [19] (as identified by TaskNav using manual checking) within the top three, five, 10, and 20 positions (see Table 2). Precision shows the percentage of relevant documents identified within a predefined list, whereas recall shows how many relevant documents were identified from all relevant ones.

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. Precision varies between 5 and 10%, with the best result being yielded by the top-10 retrieved pages. Table 2 shows that more than 50% of the relevant documentation pages were found in the top-10 retrieved positions. The percentage increases to more than 60 when the top-20 positions are considered. Overall, we could find 38 out 65 relevant documents until the top-20 in a set of 9,455 potential candidates by just analyzing the screencast transcript and ignoring the text that might appear in a scene (e.g., the source code in the IDE).

Moreover, we found that 98.8% of the API documents are below a similarity threshold of 0.12 while 55% of the relevant API documents are above the same threshold. Considering this threshold when searching for relevant API documents can help developers to find 55% of the relevant API documents in a list of 114 potential candidates from the overall corpora of 9,444 documents. Based on this results, we believe that development screencasts can be extended using API documents based on their transcript.

- By comparing only the audio transcript (the screencast transcripts but not the text that might appear in a scene, e.g. in the IDE), we could identify 38 out of the 65 relevant API documents in the first 20 positions of 9,455 candidates.
- There is a similarity threshold for relevant API documents.

5 RELATED WORK

MacLeod et al.[14] reported on the structure and content of development screencasts, as well as the different types of knowledge located in such screencasts. They studied how knowledge was presented and used axial coding extract higher-level topics. In our study, we mined Java screencasts using LDA to extract 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. [26] discussed how to link StackOverflow answers to the Java 6 SE SDK API. They used the Cosine approach to measure the similarity and LexRank to evaluate the relevance of the API documents. We extended their work by linking screencasts with API documents and by showing how similar they are.

Ponzanelli et al. [17] developed a recommender system to predict relevant YouTube videos[8] for Android development. In addition to the audio transcripts, they used an OCR tool[9] to transfers the actual video information (e.g., slides or subtitles) into text. They focus on showing relevant StackOverflow posts for random YouTube videos.

6 DISCUSSION AND LIMITATIONS

Based on the similarity analysis, we found that frames in a development screencast are much alike in contrast to other types of videos. Therefore, an identification of screencasts should be possible by using algorithms such as the Cosine similarity or LSI without knowing the actual title, tags, or the transcript of the video. Similarly, other types of videos (e.g., recorded interviews or slow motion) are also very static. We acknowledge that such approach, based

[8] http://codetube.inf.usi.ch/
[9] https://github.com/tesseract-ocr/tesseract/wiki

on frames comparison, might mistakenly find these other types of *static* videos.

The analysis of the development topics showed that development screencasts contain knowledge provided in API reference documentation. Thus, API reference documents can extend a development screencast to provide additional implementation details, making it an attractive media for those developers who do not read documentation [9]. By 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.

This preliminary study focuses on screencasts related to a specific programming language. However, there is a broad range of other development screencasts which tackle the same topics but in a different manner, or which use different programming languages with different syntax, semantics or specific tools.

The selection of the dataset might have influenced the study results. We used the title to understand the tasks performed in a development screencast, and the transcripts to understand its subtasks. Those two elements (i.e., titles and transcripts) complement each other. For example, if developers want to know how to use lists, files, and methods in a programming language like Java they might search them through an algorithm that considers the transcripts. In this way, the developers can find tasks that match the development context of interest, such as specific IDEs or libraries.

We found that UI operations—one of the most important activity performed when comprehending software [13]—are also largely performed in development screencasts. By watching screencasts, developers can understand how other developer debugged and solved similar problems.

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

Building a large dataset using the YouTube API poses some limitations since YouTube only returns a limited number of search results[10]. Thus, multiple searches, with different search terms, need to be performed. Moreover, the persistence of retrieved data is not guaranteed due to the possible deletion of the videos included in our sample.

The library we used could not completely identify and remove the verbs or stop words from the title or the transcripts. Therefore, a replication of this study could lead to different results. We recommend to use the NLTK and the pyLDAvis library to preprocess the titles and the transcripts as well as to summarize the topics of the tasks. Although different people from different countries might create developments screencasts, we did not evaluate the language quality of the screencasts which might also influence our results.

When performing a development task there is often the need for additional information to be gathered for example, from Stack Overflow, YouTube, or an API documentation. Combining all of them mean to use different types of information to perform a development task.

[10]http://stackoverflow.com/questions/25918405/youtube-api-v3-page-tokens

7 CONCLUSION AND FUTURE WORK

This paper provided a first insight on how to categorize and identify development screencasts, and how to enrich them with API documentation. We analyzed Java development screencasts on YouTube and found six frequent topics referred therein.

A software development screencast is a particular type of videos in which developers perform a tasks by focusing on relevant tools. Development screencasts are not much different from other types of screencasts. We found that frame similarity can be used to detect a development screencast on YouTube. Development screencasts can be extended by API documents to better support software developers. We found that more than half of the relevant API documents could be provided within a list of 20 items. A Cosine comparison between a screencast and a large API documentation corpus is only a preliminary, simple approach to offer developers the most relevant documents.

In the future we will focus on extracting the content of the development screencast—e.g., the code showed on the screen when using an IDE—to reach a higher precision/recall when identifying related documentation and tasks.

Further work is needed to determine the different types of knowledge [12, 14] located in screencasts to achieve a more fine-grained and precise mapping between the API reference documentation and the API elements within the IDE. This approach might require labeling every unique piece of knowledge within a screencast and use video and image features.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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).

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

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

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

[20] Terry Shepard, Margaret Lamb, and Diane Kelly. 2001. More testing should be taught. *Commun. ACM* 44, 6 (2001), 103–108.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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).

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