# CodeTube: Extracting Relevant Fragments from Software Development Video Tutorials

Luca Ponzanelli[1], Gabriele Bavota[2], Andrea Mocci[1], Massimiliano Di Penta[3]
Rocco Oliveto[4], Barbara Russo[2], Sonia Haiduc[5], Michele Lanza[1]
[1] *Università della Svizzera Italiana (USI), Switzerland* — [2] *Free University of Bozen-Bolzano, Italy*
[3] *University of Sannio, Italy* — [4] *University of Molise, Italy* — [5] *Florida State University, USA*

## ABSTRACT

Nowadays developers heavily rely on sources of informal documentation, including Q&A forums, slides, or video tutorials, the latter being particularly useful to provide introductory notions for a piece of technology. The current practice is that developers have to browse sources individually, which in the case of video tutorials is cumbersome, as they are lengthy and cannot be searched based on their contents.

We present CodeTube, a Web-based recommender system that analyzes the contents of video tutorials and is able to provide, given a query, cohesive and self-contained video fragments, along with links to relevant Stack Overflow discussions. CodeTube relies on a combination of textual analysis and image processing applied on video tutorial frames and speech transcripts to split videos into cohesive fragments, index them and identify related Stack Overflow discussions.
DEMO URL: http://codetube.inf.usi.ch
VIDEO URL: https://youtu.be/yUsUG3g87Dg

## CCS Concepts

•**Software and its engineering** → *Software maintenance tools;* Documentation;

## Keywords

Recommender Systems, Unstructured Data, Video Tutorials

## 1. INTRODUCTION

Nowadays developers rely on a variety of sources providing some kind of informal documentation. Examples include Question & Answer (Q&A) forums such as Stack Overflow, websites, and presentation slides describing a specific piece of technology. All these "informal documentation" resources complement the official documentation (*e.g.,* API reference manuals), as they can provide a step-by-step introduction to a topic, a "by example" description of how to use APIs, or in some cases a discussion from the trenches about the pros and cons of a technology, along with possible alternatives.


*ICSE '16 Companion, May 14-22, 2016, Austin, TX, USA*

DOI: http://dx.doi.org/10.1145/2889160.2889172

A special category of informal documentation is represented by video tutorials, which usually provide an introduction to a technology and illustrate its general concepts, alternating overview slides with screencasts of how a piece of software can be developed within an Integrated Development Environment (IDE). A study by MacLeod *et al.* [2] surveyed the motivation of developers for creating video tutorials, and highlighted how they allow developers to share details such as software customization knowledge, personal development experiences, implementation approaches, and application of design patterns or data structures. The study also highlighted key advantages of video tutorials compared to other resources, such as user manuals. These advantages include the ability to visually follow the changes made to the source code, to see the environment where the program is executed, to view the execution results and how they relate to the source code, and to understand a development activity at different levels of detail. In essence, video tutorials can provide a learning perspective different and complementary to that offered by traditional, text-based sources of information.

Though many sources of informal documentation exist, it is hard for developers to navigate this information and get a more integral view of a topic, since they have to browse the various sources individually, *e.g.,* starting from the ranked list of a general-purpose search engine. Moreover, when it comes to video tutorials, despite all the advantages highlighted above, their downside is that they are often lengthy, and lack support to search for specific fragments of interest. While recommender systems mining development-related information exist, to the best of our knowledge they work on a single source, *e.g.,* Stack Overflow [1, 5] or on official API documentation [8, 9], and none uses video tutorials.

We present a tool named CodeTube, which leverages the information found in video tutorials, providing developers with cohesive and self-contained video fragments relevant to a query, along with related Stack Overflow discussions. CodeTube relies on a combined analysis of (i) code constructs shown in the video frames (ii) text contained in the video frames, and (iii) voice transcripts. The fragments are then indexed based on the transcript and the textual content of each frame, and linked to relevant Stack Overflow discussions. When the user formulates a free-text query, CodeTube retrieve a ranked list of relevant video fragments, and when a fragments is selected, CodeTubealso retrieve related Stack Overflow discussions.

A detailed description of the approach behind CodeTube and its evaluation is reported in a technical paper [6].

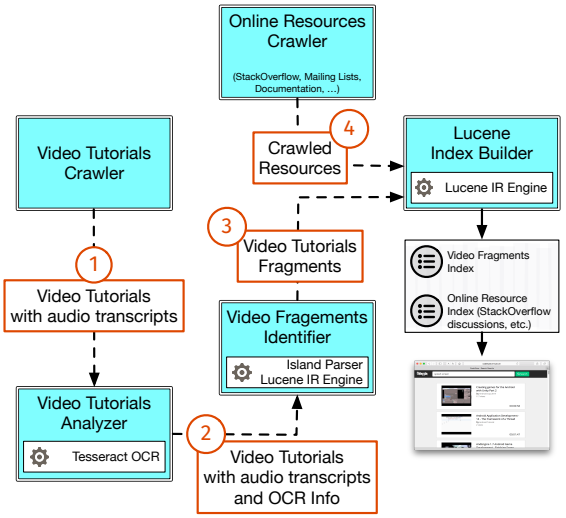


**Figure 1: CodeTube: Architecture.**

## 2. ARCHITECTURE

CodeTube is a multi-source documentation miner to locate useful pieces of information for a given task at hand. The results are fragments of video tutorials relevant to a given textual query, augmented with additional information mined from other text-based online resources.

Figure 1 depicts CodeTube's architecture. It is composed of (i) an offline analysis aimed at collecting and indexing video tutorials and other resources and implemented by the *Video Tutorials Crawler*, the *Video Tutorials Analyzer*, the *Video Fragments Identifier*, and the *Online Resources Crawler*; (ii) an online service implemented by the *Lucene Index Builder* and available through the CodeTube's Web-based GUI, where developers can search these processed resources. The analysis of video tutorials is currently limited to English videos dealing with the Java programming language.

In the following we provide basic information on what is under CodeTube's hood. Full details are available in [6].

### 2.1 Crawling and Analyzing Video Tutorials

The *Video Tutorials Crawler* is in charge of retrieving video tutorials related to a set of queries $Q$ the user is interested in (*e.g.,* "Android development"). Each query in $Q$ is run by the crawler using the YouTube Data API[1] to get the list of YouTube channels relevant to the given query $q_i \in Q$. For each channel the *Video Tutorials Crawler* retrieves the metadata (*e.g.,* video URL, title, description) and the audio transcripts, which are either automatically generated or written by the author of the tutorial. Once the videos have been crawled, their metadata is provided as input to the *Video Tutorial Analyzer* (step 1 in Figure 1), which extracts pieces of information to isolate video fragments related to a specific topic. Its goal is to characterize each video frame with the text and the source code it contains. The text is extracted using the Optical Character Recognition (OCR) tool tesseract-ocr[2]. The source code shown in the frame (if any) is identified using an island parser [3, 7] on the text extracted by the OCR.

[1]https://developers.google.com/youtube/v3/
[2]https://github.com/tesseract-ocr

Information about the audio transcript, the text contained in each frame, and the code constructs (if any) each frame contains is provided as input to the *Video Fragments Identifier* (step 2 in Figure 1), which is in charge of detecting cohesive fragments in a video tutorial. It starts by identifying video fragments characterized by the presence of a specific piece of code. The conjecture is that a frame containing a code snippet is coupled to the surrounding video frames showing (parts of) the same code. Identifying the video frames containing a specific (*i.e.,* the same) code snippet presents non-trivial challenges. For example, a piece of code could be written incrementally during a video tutorial leading frames related to the same code snippet to show different portions of it. Such challenges are overcome by exploiting a set of heuristics [6], which allows CodeTube to identify code intervals (*i.e.,* parts of the video tutorial related to the same code snippet). The code intervals are subsequently refined by identifying the audio transcripts starting and/or ending inside each code interval. CodeTube uses the beginning of the first and the end of the last relevant audio transcript sentence for a code interval to extend its duration and avoid that the interval starts or ends with a broken sentence. The extended code interval represents an identified video fragment.

Non-code frames in the video that have not been assigned to any video fragment are merged to semantically related fragments in the vicinity. CodeTube merges two subsequent fragments if one of two conditions applies: their textual similarity is higher than $\beta$, or one of the two frames is shorter than $\gamma$ seconds (to avoid the extraction of very short fragments). Both thresholds have been tuned, showing that the best results can be achieved with $\beta$=15% and $\gamma$=50s [6].

### 2.2 Integrating Other Sources of Information

CodeTube can be enriched by mining other online resources. Our long-term goal [4] is to offer a *holistic* view of all available information, as we argue that no single type of resource can offer exhaustive assistance. To exemplify this, we added Stack Overflow as additional information source. We mined and extracted discussions from the Stack Overflow public dump[3] of March 2015 by considering the <android> tag. This is implemented in the *Online Resources Crawler* (Figure 1). Then, we indexed both the extracted video fragments (step 3 in Figure 1) and the Stack Overflow discussions (step 4 in Figure 1), using the Lucene Index Builder[4], where each video fragment, and each post of Stack Overflow are considered as documents. The text indexed for a video fragment is represented by the terms contained in its frames and audio transcripts. The text indexed for the Stack Overflow posts is represented by the terms they contain.

Currently, CodeTube is available with 4,747 indexed videos related to Android development extracted from YouTube. This resulted in 38,783 video fragments complemented by 638,756 indexed Stack Overflow discussions tagged with "android". The indexed information is made available in the CodeTube frontend, detailed next.

## 3. CODETUBE IN ACTION

Alice is working on her first Android app. Her app is meant to mark points of interest (*e.g.,* restaurants, hotels, *etc.*) close to the smartphone location provided by the integrated

[3]https://archive.org/details/stackexchange
[4]https://lucene.apache.org/

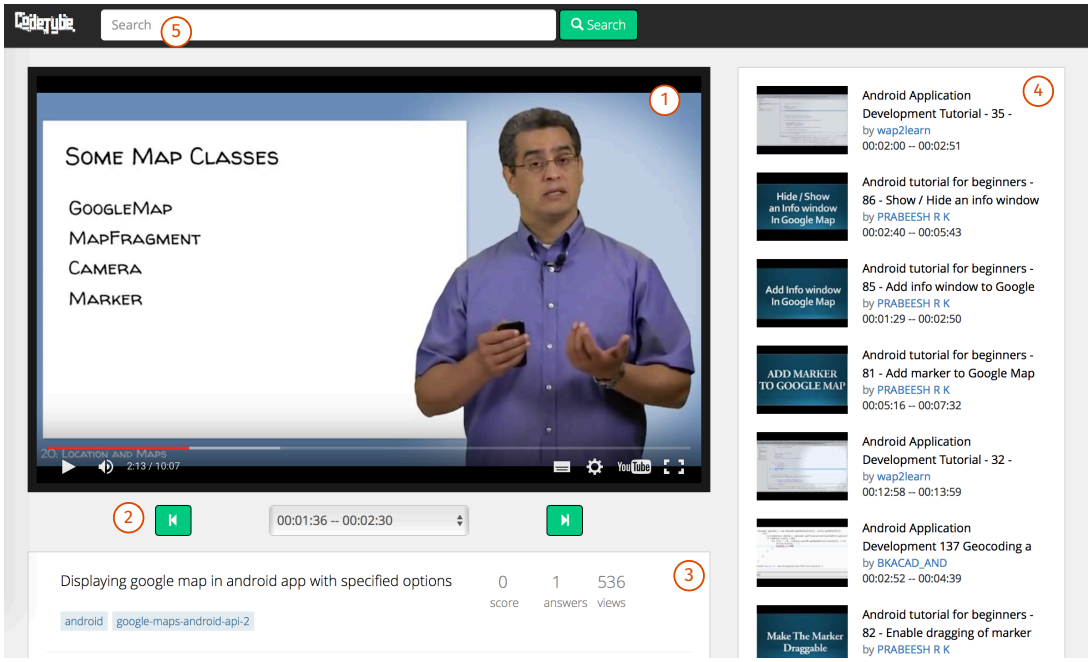


**Figure 2: CodeTube: Main GUI.**

GPS. Alice has little knowledge about the development of Android apps but, after some hours of work, she manages to visualize on screen a Google map showing the area nearby the smartphone location. The next step is to find a way to mark points of interest.

Alice uses CodeTube to formulate the query "*pin a marker on google maps*" (see Figure 3).

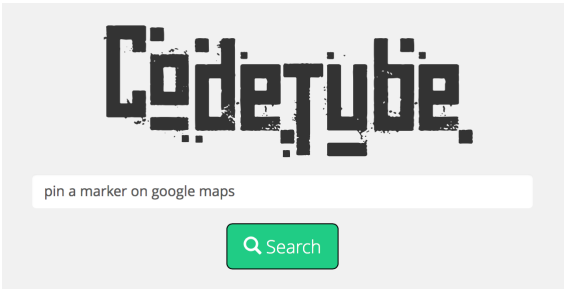


**Figure 3: CodeTube: Search Field.**

Alice's query is run against the indexed video fragments and a list of relevant results is presented to her (see Figure 4 – the list of relevant fragments is cut to the first two results for the sake of space).

Alice selects the first video fragment and accesses the main CodeTube GUI depicted in Figure 2. CodeTube uses the YouTube player Figure 2-(1) provided by the YouTube API[5] to play the selected video fragment. The video starts at the time devised by the selected fragment. CodeTube provides an additional controller (2) to visualize the timestamps of the fragments identified by our approach, to select a specific fragment, or to move to the next/previous fragment. During the video playback, the selector underneath the video player

[5]https://developers.google.com/youtube/js_api_reference

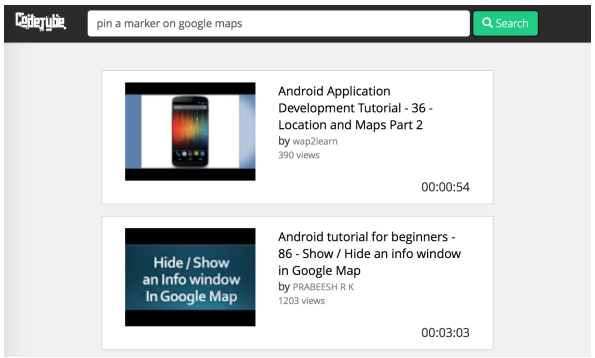


**Figure 4: CodeTube: Relevant Video Fragments.**

keeps the pace of the video timing and shows the current fragment.

When a new fragment is reached, or the user jumps to it, CodeTube automatically extracts a query from the text contained in the fragment (*i.e.,* transcripts and OCR output of the frames it contains), queries both the index of Stack Overflow and of the video fragments, and updates the related discussions (3) and YouTube video fragments (4). The search bar (5) is always available to the user to run new queries.

Alice looks at the video fragment and gathers some conceptual knowledge about the Android API classes implementing services related to Google maps (see Figure 4). Then, she decides to look for a more practical tutorial, actually showing how to pin a marker on Google maps. She moves her attention towards the list of related video fragments shown on the left side of the screen—Figure 2-(4)—and clicks on the second one[6]. The tutorial, and in particular the specific video fragment, shows working code implementing the

[6]http://tinyurl.com/nmcjw85

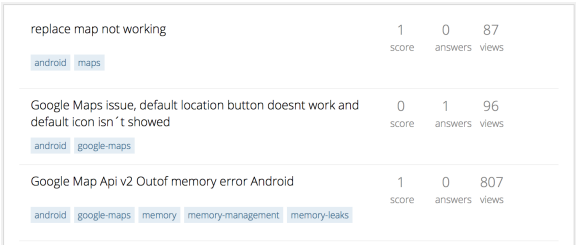


**Figure 5: CodeTube: Relevant Stack Overflow Discussions.**

feature Alice is interested in. Moreover, Alice exploits the related Stack Overflow discussions that are shown below the YouTube player (the top three are shown in Figure 5).

These discussions deal with issues she could encounter while working with Google maps on an Android device. Alice thus starts inspecting the discussions.

# 4. EVALUATION SUMMARY

CodeTube has been evaluated through two studies [6].

In the first study, 34 developers with Android experience performed an intrinsic evaluation of the results produced by CodeTube through an online survey. The participants evaluated (i) the coherence and conciseness of the video fragments produced by CodeTube, as well as their relevance to a query, as compared to the results returned by YouTube, and (ii) the relevance and complementarity of Stack Overflow discussions returned by CodeTube for specific video fragments. Almost 80% of video fragments generated by CodeTube were considered *cohesive* by the study participants, while ∼60% were also judged self-contained and only ∼10% were considered not self-contained (there was disagreement for the remaining 30%). Only 20% of the Stack Overflow discussions retrieved by CodeTube were considered unrelated to the corresponding video fragment and 82% of them were judged as complementary (in terms of provided information) to the video fragment. Finally, in terms of relevance of the retrieved fragments for a given query, CodeTube performed as well as YouTube, showing comparable retrieval performances.

In the second study, we performed an extrinsic evaluation of the approach by introducing CodeTube to three leading developers involved in the development of Android apps. We asked them questions about the usefulness of CodeTube, focusing on the value of extracting fragments from video tutorials, and of providing recommendations by combining different sources of information. The reception of CodeTube was positive: all leading developers saw great value and even greater potential in this line of work.

More details about both evaluations are available in the techical paper describing CodeTube's approach and its evaluation [6].

# 5. CONCLUSION

We presented CodeTube, a Web-based recommender that, given a query provided by the user, suggests cohesive and self-contained fragments from video tutorials, and complements them with relevant Stack Overflow discussions.

CodeTube is able to mine video tutorials by analyzing both the contents of video frames and speech transcripts, splitting long video tutorials into coherent and cohesive fragments. The analysis effectively combines different image processing techniques, like shape detection and OCR, and text analyses, such as island parsing.

CodeTube is, to our knowledge, the first freely available recommender that supports video tutorials for software engineering and complements them with other resources like Stack Overflow. We also briefly presented the results of the evaluation of CodeTube, which showed a positive reception from developers, both in its actual status and its potential to become an effective *holistic recommender*.

**Acknowledgements.** The authors would like to thank all the people involved in the evaluation of CodeTube. Ponzanelli and Lanza thank the Swiss National Science foundation for the financial support through SNF Project "ESSENTIALS", No. 153129. Bavota is supported by the Free University of Bozen-Bolzano through the STREAM project (IN2036). Haiduc is supported in part by the National Science Foundation grant CCF-1526929.

# 6. REFERENCES

[1] R. Holmes and G. C. Murphy. Using structural context to recommend source code examples. In *Proceedings of ICSE 2005 (27th International Conference on Software Engineering)*, pages 117–125. ACM, 2005.

[2] L. MacLeod, M.-A. Storey, and A. Bergen. Code, camera, action: How software developers document and share program knowledge using YouTube. In *Proceedings of ICPC 2015 (23rd IEEE International Conference on Program Comprehension)*, 2015.

[3] L. Moonen. Generating robust parsers using island grammars. In *Proceedings of WCRE 2001 (8th Working Conference on Reverse Engineering)*, pages 13–22. IEEE CS, 2001.

[4] L. Ponzanelli. Holistic recommender systems for software engineering. In *Proceedings of ICSE 2014 (36th ACM/IEEE International Conference on Software Engineering), Doctoral Symposium*, pages 686–689. ACM, 2014.

[5] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, and M. Lanza. Mining StackOverflow to turn the IDE into a self-confident programming Prompter. In *Proceedings of MSR 2014 (11th Working Conference on Mining Software Repositories)*, pages 102–111. ACM Press, 2014.

[6] L. Ponzanelli, G. Bavota, A. Mocci, M. Di Penta, R. Oliveto, M. Hasan, B. Russo, S. Haiduc, and M. Lanza. Too long; didn't watch! extracting relevant fragments from software development video tutorials. Technical report, Proceedings of the ICSE 2016 (38th ACM-IEEE International Conference on Software Engineering), 2015.

[7] L. Ponzanelli, A. Mocci, and M. Lanza. Stormed: Stack overflow ready made data. In *Proceedings of MSR 2015 (12th Working Conference on Mining Software Repositories)*, pages 474–477. ACM Press, 2015.

[8] P. C. Rigby and M. P. Robillard. Discovering essential code elements in informal documentation. In *Proceedings of ICSE 2013 (35th International Conference on Software Engineering)*, pages 832–841. IEEE Press, 2013.

[9] M. P. Robillard and Y. B. Chhetri. Recommending reference API documentation. *Empirical Software Engineering*, pages 1–29, 2014.