Summarizing Relevant Parts from Technical Videos

Mahmood Vahedi*, Mohammad Masudur Rahman†, Foutse Khomh*, Gias Uddin‡, Giuliano Antoniol*
*Ecole Polytechnique de Montreal, †Dalhousie University, ‡University of Calgary
{mahmood.vahedi, foutse.khomh, giuliano.antoniol}@polymtl.ca, masud.rahman@dal.ca, gias.uddin@ucalgary.ca

***Abstract*—Software developers frequently watch technical videos and tutorials online as solutions to their problems. However, the audiovisual explanations of the videos might also claim more time from the developers than the text-only materials (e.g., programming Q&A threads). Thus, pinpointing and summarizing the relevant fragments from these videos could save the developers valuable time and effort. In this paper, we propose a novel technique – TechTube – that can be used to find video segments that are relevant to a given technical task. TechTube allows a developer to express the task as a natural language query. To account for missing vocabularies in the query, TechTube automatically reformulates the query using techniques based on information retrieval. The reformulated query is matched against a repository of online technical videos. The output from TechTube is a sequence of *relevant* video segments that can be useful to implement the task at hand. Unlike previous researches, our approach splits the video by detecting silence in video audio tracks. Experiments using 98 programming related search queries show that our approach delivers the relevant videos within the Top-5 results 93% of the time with a mean average precision of 76%. We also find that TechTube can deliver the most relevant section of a technical video with 67% precision and 53% recall that outperforms the closely related existing approach from the literature. Our developer study involving 16 participants reports that they found the video summaries generated by TechTube very accurate, precise, concise, and very useful for their programming tasks rather than the original complete videos.**

***Keywords*—Software engineering, Query Reformulation, Document Information Retrieval, Text Similarity, Video Summarizing**

I. Introduction

Studies suggest that software developers spend about 20% of their development time in searching for programming solutions on the Internet [1]. Besides programming Q&A websites (e.g., Stack Overflow) and software repositories (e.g., GitHub, SourceForge), they frequently look for relevant online technical videos. YouTube is one of the most popular websites on the Internet that offers millions of high-quality technical videos, including programming tutorials [2]. Video tutorials capture the finer details of a programming task through audiovisual information, annotations, and also non-verbal cues. Thus, programming video tutorials are often a better and faster choice for the developers than the traditional text-only resources (e.g., API documentation) [3]. However, these video tutorials might also claim significant time and effort from the developers during problem-solving. Unlike the programming Q&A threads, programming videos might not be fully indexed or well organized *e.g.,* structured as explicit questions and answers. They might also contain noisy, redundant information that might be of little interest to the developers [4]. Thus, the developers often attempt to find out the relevant sections by skimming (i.e., forward and backward the video stream) through the videos. However, skimming through a video is often more time-consuming than scrolling through a regular text-based website. Thus, pinpointing the relevant sections within a technical video and guiding developers to these sections has the potential to save their valuable time and effort.

There have been a few existing studies [4, 5, 6] that attempt to extract important sections from online technical videos. Adcock et al. [5] capture keywords from the slides of a webcast using Optical Character Recognition (OCR) and then help locate the important slides using text processing and Information Retrieval methods (e.g., Lucene). Since their approach heavily relies on OCR technology to collect the textual contents, it could not perform well with low-quality or noisy videos. Ponzanelli et al. [4] suggest relevant sections from YouTube videos where they use textual contents from the video frames containing code segments to split their videos. Thus, their approach could fail to split the technical videos that do not contain any code segments (e.g., tool installations, webinars). Furthermore, their algorithms and heuristics that are designed to detect the Java code might not be suitable for other programming languages (e.g., Python).

In this paper, we propose a novel approach – TechTube – that identifies the most relevant sections from a technical video and delivers coherent, concise, and relevant video summaries against a natural language query. TechTube allows a developer to express a query, representing the task at hand, in natural language. To account for missing terms and improve recall, our approach automatically augments the query by reformulating it with popular reformulation techniques (e.g., Rocchio [7]). The reformulated query is then matched against a repository of online technical videos. The repository can be populated with videos from any available online site. The output from TechTube is a sequence of *relevant* video segments (a.k.a., video summary) that can be useful to implement the task at hand. Such a video summary can help the developers gather sought information in less time and with reduced information overload. Unlike Adcock et al., our approach does not rely on OCR that could be subject to poor-quality or noisy videos. TechTube differs also from Ponzanelli et al., as it does not rely on code segments for video splitting and is not restricted to Java-only programming videos. We implemented TechTube as a web-based tool that is easy to integrate with any technical video websites (e.g., YouTube).

We evaluate our approach that provides relevant video summaries from technical videos in two ways: (a) empirical evaluation (RQ_1–RQ_4) and (b) developer study (RQ_5). First, we select 98 programming related search queries from an *existing benchmark* [8] and Stack Overflow Q&A website. We manually build the *ground truth* (relevant videos plus relevant video sections) for these queries by involving three graduate-level, Software Engineering students. Then we use this ground truth to evaluate how our approach performs in retrieving the relevant technical videos. Our experiments show that TechTube, 93% of the time, delivers the relevant videos within the Top-5 results with a mean average precision of 76%. We also evaluate the extracted video sections against the ground truth and find that TechTube delivers the most relevant section from a technical video with a precision of 67% and a recall of 53%, which outperforms the CodeTube approach [4] with 6% higher precision, more than 200% higher recall, and 100% higher F1-score. Second, to demonstrate the benefits of our approach, we conduct a developer study in two phases. In the first phase, we ask 16 participants (7 professional developers + 9 graduate level students) to perform six programming tasks of three difficulty levels (*easy, medium, hard*) using TechTube. In the second phase, we collect their comparative feedback on the accuracy, preciseness, conciseness, and usefulness of TechTube summaries against the similar original complete videos. Most noticeably, all participants were able to successfully implement their programming tasks using TechTube. Furthermore, they found the summarized videos provided by TechTube to be *very accurate*, *precise*, *concise* and *very useful* according to a Likert scale (RQ_5).

Thus, our paper makes the following contributions:

(a) A novel approach to identify the most relevant section of a technical video to a given natural language query.
(b) Comprehensive evaluation of the approach using 98 search queries, comparison with a closely related approach from the state of the art [4] and a developer study involving 16 participants.
(c) A replication package [9] of our approach – TechTube – for the replication and third-party reuse.

II. Background

This section introduces key concepts used in the paper.
Audio segmenting techniques attempt to identify the change points within an audio file that could be useful for various applications such as monitoring and summarizing a group meeting, and indexing a broadcast news [10, 11, 12]. One of these techniques is metric-based audio segmentation. The technique identifies the maximum distance between two neighbouring windows and captures a speaker's silence. Then it segments the audio file using the timestamp of the silence [10]. We use this speaker's silence-based method for audio/video segmenting in our approach (Section III-A1).

Query Reformulation: To find relevant code examples, developers often use typical queries in the form of natural language texts. Bajracharya and Lopes [13] report that only 12% of these queries lead to relevant results due to vocabulary mismatch problems [14, 15]. Existing researches [16, 17, 18] show that the developers reformulate their issued queries on their own around 33%-73% of the time. However, they can make mistakes during this reformulation and thus it can cost more time and efforts [18]. Automated query reformulation aids the developers to save their time and efforts. Adding keywords from a relevant previous result may help the developers mitigate such issues [19]. There have been a number of query expansion techniques using information retrieval methods [20]. We use pseudo-relevance feedback and Term Frequency (TF) [21] to reformulate a query in TechTube, which has been widely used to expand the search queries [22]. In particular, we carefully select the most frequent keywords from the audio contents of technical videos and reformulate a query at hand for retrieving the relevant videos and video summaries (Section III-B2).

Text Similarity: Text similarity measures are popular among the researchers and developers due to their high effectiveness in various tasks such as text classification, document clustering, and text summarization [23, 24]. There have been many text-similarity approaches, such as string-based similarity, character-based similarity, corpus-based similarity, and term-based similarity [23]. In our proposed TechTube engine, we offer text similarity support using both the traditional cosine similarity measure [25, 26] as well as more advanced BM25 algorithm [27]. Cosine similarity computes the cosine value of the angle between two vectorized texts [25, 26]. BM25 algorithm is based on an adaptation of the popular Inverse Document Frequency (IDF) algorithm [28]. Both algorithms use the vectorized representation of the textual contents in input query and the target documents (that are subject to search) to compute the similarity (Section III-B3).

III. TechTube

Figure 1 shows the schematic diagram of TechTube, our technical video summarization framework. The framework is composed of an offline component and an online component. The offline component is used to download and preprocess technical videos from online resources. The online component offers a search and summarization engine of the downloaded videos to a developer. We describe the two components below.

A. TechTube Offline Component

We use TechTube offline component to download technical videos and their metadata from online resources. We preprocess the videos into *informative* and *isolated* video segments. The video segments and their metadata are stored in a database. TechTube online component uses the database to present the summarized video segments to the developer.

The **Video Audio Processor** module preprocesses a downloaded video as follows: (1) Audio Chunking: we segment the audio into different chunks. (2) Text Extraction: we extract the speech text for each of those segmented chunks.

1) Video Chunking: The input to this step is a downloaded video. The output is a list of video *chunks*, *i.e.,* sequence of contiguous video frames. One of the practical features

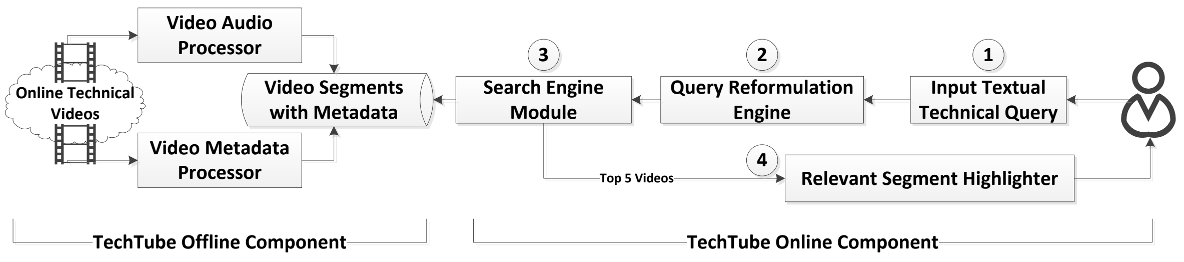
Video Audio Processor
Online Technical Videos
Video Metadata Processor
Video Segments with Metadata
3
Search Engine Module
2
Query Reformulation Engine
1
Input Textual Technical Query
4
Top 5 Videos
Relevant Segment Highlighter
TechTube Offline Component
TechTube Online Component

Figure 1. The schematic diagram of our proposed TechTube framework

of each tutorial video is the lecturer's voice; the descriptive support for the image frames playing sequentially. Human uses prosody to stress and highlight speech points: silences are an essential prosody feature. Speakers, often, put a long silence when they change the subject or underline a very relevant point. TechTube uses speaker's silences, the prosody element, to segment utterances. To detect pauses in audio, TechTube uses the amplitude in a given audio file. An amplitude value less than a given threshold is considered silence. Given that different video files can be recorded in different settings, their amplitudes can differ from each other. For example, a speaker can have background music, while another may not have any such background music during recording. Therefore, we dynamically determine the silence per video file as follows. We compute the amplitude of each frame from the audio of a video. We compute the average of the amplitudes. If, for a given frame, the amplitude value drops below the average amplitude value across the frames, we consider that as a silence/pause. Based on the above detection of silence in the audio, we can then segment an audio file into multiple chunks. That means, two consecutive chunks are separated by a pause/silence. We create a timestamp for each chunk to specify the starting time and finishing time of the chunks based on their time of play on the video. For video chunking, the TechTube framework currently supports a cross-platform application called FFMPEG [29]. The tool is used to extract the audio and video parts out of an input video file. To detect silences in the audio, TechTube currently uses PyDub [30].

2) Speech Recognition: The input to this step is a video chunk with its associated audio. The output is a transcribed text out of the audio. TechTube currently uses the Google speech recognition engine [31] to write the speech of each audio chunk into a .txt file for similarity comparison and further usages. We select the Google speech engine because it is a continuous speech recognition engine, it is speaker-independent, plus it scales up to millions of videos. In our manual evaluation of the transcribe texts, we found that the recognition accuracy of the engine was 90% (average).

B. TechTube Online Component

The online component offers an interface to a developer to search the downloaded videos using a natural language query. Given as input a textual query related to a technical task, TechTube uses a query reformulation method to find the appropriate videos by mining their metadata, such as tags and video descriptions (if applicable). TechTube measures the similarity rate of the reformulated query and the extracted speech texts using the text-similarity algorithms. For each of the top query results, TechTube then shows the relevant fragment of the video in a summarized interface. Specifically, the online component is composed of four modules:

1) *Input Query Processor* provides a search box where the developer can input a textual query,
2) *Query Reformulation Engine* automatically expands the input query to make it more nuanced and accurate,
3) *Search Engine Module* matches the reformulated query against the preprocessed videos in the database,
4) *Relevant Segment Highlighter* picks the top five videos and produces a summarized version of those videos by highlighting video segments relating to the input query.

We describe the four modules below.

1) Input Textual Technical Query: TechTube allows a developer to describe a technical task using natural language in a search box. One of the main parts of search tasks that impact the relevancy rate of the results is the generated search query. A well-dictated non-noisy rich query leads to gaining more accurate outcomes. Due to this issue, TechTube uses a necessary query cleaning and pre-processing method for text normalization, which is a combination of stemming, lemmatization, and stopword removal [19].

2) Query Reformulation Engine: The textual description of a task (i.e., the query) provided by a developer may not contain all the necessary keywords to describe the task properly. To address this problem, TechTube reformulates the query using a standard query expansion technique – Rocchio's expansion [7]. TechTube uses the technique as follows. Each downloaded video in our database contains metadata (e.g., title, tags, description) besides their subtitles (using auto-generated subtitles from YouTube) and audio speech that explain its discussion topics. Our approach first captures pseudo-relevance feedback [32] on a developer's query by executing the query against all the available videos. We pick the topmost retrieved videos from the query results. The underlying idea is that these somewhat relevant videos might contain essential keywords that could complement the developer's free-form query.

TechTube then finds all the keywords in the feedback videos that we collected using the TechTube offline component. To get the keywords, TechTube first preprocesses the video texts using standard natural language processing (e.g., stopword removal, lemmatization) and then picks the most frequent keywords from these feedback videos. These keywords are then used to expand/reformulate the search query issued by the developer. A similar approach has been adopted by many relevant studies from the literature [22, 33, 34].

3) Search Engine Module: This module is responsible for two tasks. First, it detects four most relevant videos given as input the reformulated query. Second, for each returned relevant video, the module further detects chunks in the video that are relevant to the query. The search engine module utilizes Apache Lucene [35]. We picked Apache Lucene because it is a popular open-source search engine software library and it is regularly used in state of the art query reformulation research in Software Engineering (see [36]). To compute the similarity between the input query and a video, TechTube takes into account the speech text of the video. TechTube produces a vectorized representation of the input query and the speech text. To produce the vectors, TechTube applies standard text normalization approaches, such as stopword removal and lemmatization. The search module now offers two similarity metrics to compute the similarity between the input query and the speech text: Cosine Similarity [28] and BM25 [27]. Upon computation of the similarity, TechTube sorts the videos by the similarity values. TechTube takes the top four videos with the most similarity values. For each of the four videos, TechTube then computes the similarity between the input query and each chunk in the video.

4) Relevant Segment Highlighter: At this stage, for each relevant video identified by the Search Engine Module, TechTube has several, non-contiguous, sequences of similar chunks. To ensure a coherent and smooth documentation experience from the video given the input query, TechTube needs to combine similar chunks. However, it may happen that two very small relevant chunks are separated by a pause. In such cases, the chunks may be relevant, but they cannot provide a coherent experience. Therefore, to elect the most relevant video fragment (*i.e.,* non-contiguous chunks sequence), TechTube uses the Longest Common Subsequences (a.k.a., LCS) [37] strategy. Figure 2 explains the strategy as follows. The black bars are relevant video chunks. The white bars are less relevant video chunks and the spaces between sequences denote pauses in the audio. Each bar denotes a video chunk. The length of a bar corresponds to its similarity to the input query, i.e., a longer bar is more relevant. To produce a coherent representation of the relevant video chunks out of an input video, TechTube starts with the first video chunk that is similar to the query. To reduce noise, TechTube only considers a video chunk with a similarity higher than a threshold (half similarity of the highest similar chunk). TechTube then proceeds to the next video chunk. If the chunk is relevant, it is included. If several consecutive chunks are found to be relevant, they form a sequence. This approach stops when a chunk is found to have a pause, or it is relevant less than the threshold. For example, in Figure 2, the first LCS has a total of seven consecutive video chunks. The second LCS has a total of six consecutive video chunks. To offer a uniform and coherent documentation experience out of the video, TechTube picks all the video chunks between LCS1 and LCS2 (i.e., all including the pauses in between). Using this same approach, the two consecutive relevant video chunks after LCS2 are not included, because they are significantly smaller in length than the previous two longest common subsequences (i.e., 7 for LCS1 and 6 for LCS2). We apply this process to each of the five selected videos. Each video automatically starts at the beginning of the first LCS in the video. Upon the finish of the first LCS, the user is taken to the second LCS of the video. If a user wants to watch the original videos, they can ignore the timing of the video and start the video manually from the beginning time or any other time.

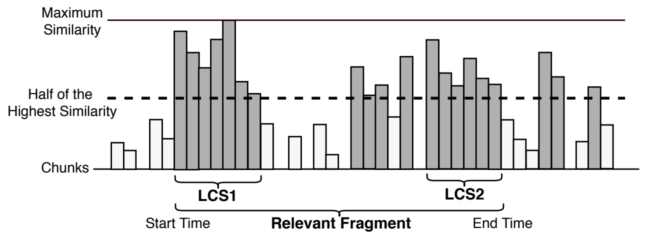

Figure 2. TechTube Relevant Fragment Identification

IV. EXPERIMENTS

We evaluate TechTube in two different ways: (1) by means of an empirical evaluation and (2) by a user study, a case study, involving 16 developers. The first empirical evaluation focuses on TechTube's effectiveness to retrieve relevant videos and to produce meaningful video summaries. In particular, we examine the roles of three TechTube components, namely – retrieval technique used in the Search Engine Module (e.g., Apache Lucene), video splitting method, and query reformulation mechanism. In a second step, we conduct a user study involving 16 professional developers and investigate whether TechTube provides relevant, concise, useful video summaries against their search queries. Overall, we address the five research questions. While RQ_1 to RQ_4 deal with our empirical evaluation, RQ_5 deals with the developer study.

RQ1. How does TechTube perform in (a) retrieving the relevant technical videos against textual queries, and (b) identifying the most relevant sections from the videos?

RQ2. How does Apache Lucene perform in retrieving the relevant videos? Is its use justified?

RQ3. Is the use of silence-based video splitting justified?

RQ4. Does the query reformulation improve the retrieval (a) of relevant video and (b) relevant fragment of a video?

RQ5. Does TechTube deliver relevant, concise, useful video contents against the developers' queries?

A. Experimental Dataset

We collect a total of 98 natural language queries related to Online Repository (e.g. Github) maintenance, Java and Python programming tasks for our experiments due to their popularity among developers. The queries related to Java programming were taken from an *existing benchmark* [8] whereas the rest of the queries were extracted from the top-scored Q&A threads of Stack Overflow considering the post topic as the query. We then populate our database by downloading 400 technical videos from YouTube that were retrieved using the keywords from these 98 queries and that looked relevant to the queries at the first glance. Our inclusion criteria were: (1) each video was longer than three minutes (three minutes videos can be considered as a video summary), and (2) the video contained English speech track. Since some of our queries were similar in the concept, we could use multiple queries against one video, so we extract 98 videos out of 400 videos to have one video per query. The human filtering of the videos leads our experimental dataset to contain different types of videos by the aspect of time duration, different visual options (low quality frames, high quality frames, slide presentation frames, code development frames and etc.) and the videos that provides multiple topics in one video. We also collect the metadata (e.g., tags, title, description) and the subtitles (i.e., textual narrations generated automatically by YouTube) for each of the videos for our experiments. We construct a *ground truth* (i.e., relevant videos plus segmented relevant video sections) database for the 98 queries as follows. We first segment each video manually. Then three software engineering graduate students who do not have any clues about the approach manually analyzed the videos and the segments to assess their relevancy to the 98 queries. They first labelled 10 videos and consult together to have a similar taste of labelling and then they labelled their share of videos. About 50 man-hours were spent to construct the ground-truth. We use this ground truth database to answer RQ_1 – RQ_4.

Our experimental dataset and replication package can be found online [9] for the replication and third-party reuse.

B. RQ_1: Performance of TechTube in Retrieving Relevant Video and Relevant Video Segments

First, we evaluate the performance of TechTube to retrieve relevant videos given for an input query. We compute the similarity of the relevant videos using the BM25 metric in TechTube search engine module. Second, we evaluate the performance of TechTube to retrieve relevant video segments from a video and compare the output with CodeTube [4], a closely related existing approach.

We run each of our 98 queries against TechTube. For each query, we pick the top-5 results by keeping their ranks as returned by TechTube. For each query, we then compare the results against our ground truth database using three performance metrics – Hit@K, MAP and MRR. Table I reports the results. Once relevant videos are retrieved, we also extract the relevant video sections (a.k.a., video summary) using both approaches (i.e. TechTube and CodeTube). We compare the relevant video segments of each approach against our *ground truth* database by computing three other metrics –precision, recall, and F-score. Fig. 3 reports the obtained results.

Table I
TECHTUBE'S PERFORMANCE IN VIDEO RETRIEVAL

Performance Metric	**Top-1**	**Top-3**	**Top-5**
Top-K Accuracy/Hit@K	64.28%	90.81%	92.85%
Mean Reciprocal Rank (MRR)	-	-	0.76
Mean Average Precision@K (MAP)	-	-	76.13%

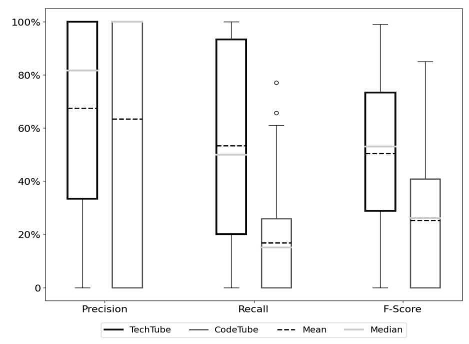

Figure 3. TechTube's Performance in the retrieval of relevant video sections

From Table I, we see that TechTube retrieves the relevant videos for 93% of the 98 search queries within their Top-5 result. The mean average precision is 76%. That is, TechTube retrieves a relevant technical video for 9 out of 10 queries. It also achieves a mean reciprocal rank of 0.76, which indicates that, on average, the most relevant videos can be found at the first or second position within the ranked retrieved results. Table I shows that TechTube delivers the most relevant videos at the topmost position (i.e., Hit@1) for 64% of the queries.

Based on Fig.3, on average, TechTube can identify the most relevant video fragments for a search query with a precision of 67%, a recall of 53% and an F-score of 50% while CodeTube [4] can identify the most relevant video section for a search query with a precision of 64%, a recall of 17% and an F-score of 25%. In total, our approach – TechTube – outperformes the CodeTube approach by 6% higher precision, 230% higher recall, and 100% higher F1-Score. Table II shows the detail of the comparison between TechTube and CodeTube approaches against a variety of videos such as videos related to Java programming, Python programming and videos with long duration (i.e. more than 15 minutes) and the videos with duration of less than 15 minutes. On Table II we see that, TechTube delivers relevant sections for the Python related video tutorials with 62% precision which outperforms the CodeTube approach by 13% higher precision, 180% higher recall, and 140% higher F1-score. Due to the use of Java based filtering and the use of island parser to find the Java constructors, CodeTube provides a weaker performance on Python related video tutorials.

On the same table, we see that TechTube deals better with the videos with duration time of more than 15 minutes than CodeTube. On this type of videos, TechTube provides almost 40% higher precision, 500% higher recall, and 250% higher F1-score. Because these long videos have a high number of dissimilar video frames, CodeTube's island parser fails in identifying code segments and creates many irrelevant segments, which is likely the reason behind its poor performance.

Table II
TECHTUBE'S PERFORMANCE IN VIDEO SECTION RETRIEVAL

Approach	Videos	Precision	Recall	F1-Score
TechTube	all	**67.46%**	**53.44%**	**50.47%**
	Java	71.53%	50.68%	48.12%
	Python	62.68%	54.74%	51.31%
	more than 15 minutes	39.21%	49.00%	35.94%
	less than 15 minutes	74.43%	54.53%	54.05%
CodeTube	all	63.42%	16.85%	25.23%
	Java	69.50%	19.32%	27.99%
	Python	55.78%	14.00%	21.87%
	more than 15 minutes	28.82%	8.22%	10.80%
	less than 15 minutes	71.96%	18.98%	28.80%

TechTube analyzes the speech and audio of the videos files and measures the textual similarities. In contrast, CodeTube relies on the visual features (e.g., video frames, OCR extracted content and etc.) of the videos. Accordingly, our approach performs better in variety of video types with weak visual features (e.g., low quality video, zooming in and zooming out of a frame, scrolling and etc).

Summary of RQ$_1$: TechTube can retrieve relevant technical videos for **93%** of the queries with **76%** precision. It also can deliver the relevant sections from these videos (a.k.a., video summary) with 67% precision which is 6% higher than CodeTube, 53% recall which is more than 200% higher than CodeTube, and 50% F1-Score which is 100% higher than the CodeTube approach.

C. RQ$_2$: Impact of Video Search Engine in TechTube

TechTube employs a search engine module (Step 3, Fig. 1) accepting as input a natural language query and returning relevant videos. We use two different methods – *ad hoc* and *Apache Lucene* – for retrieving the videos where each video document is represented with its metadata (e.g., tags) and subtitles collected from YouTube. The ad hoc method relies on *cosine similarity* algorithm [26] whereas Lucene employs a sophisticated Information Retrieval technique called BM25 [27]. We experiment with both search engines – ad hoc and Lucene, and compare their performance in the relevant video retrieval using three performance metrics – Hit@K, MAP and MRR. Table III summarizes our comparative analysis. We thus answer RQ$_2$ as follows.

Table III
ROLE OF SEARCH ENGINE IN RELEVANT VIDEO RETRIEVAL

Technique	Document	Hit@1	Hit@3	Hit@5	MAP	MRR
Ad-hoc	ST	62.24%	78.57%	82.65%	70.76%	0.71
	T	62.24%	88.77%	**91.83%**	**75.03%**	**.75**
	{ST + T}	63.26%	73.46%	80.61%	69.51%	0.70
Lucene	ST	63.26%	78.57%	82.65%	72.02%	0.72
	T	53.06%	77.55%	78.57%	64.2%	0.64
	{ST + T}	64.28%	90.81%	**92.85%**	**76.14%**	**0.76**

ST = Subtitles, **T** = Tags

Table III, shows that both search engines –ad hoc and Lucene– perform well in retrieving the relevant videos. We represent each video document as different combinations of its tags and subtitles and then investigate how these two search engines can retrieve the relevant technical videos.

When subtitles are used as a proxy to video document, both search engines achieve about 83% Hit@5 with ≈72% mean average precision, which are promising. Subtitles capture the discussed topics of a video in details that can be analysed to determine the relevance. When tags are used as a proxy to video document, the ad hoc method achieves 17% higher Hit@5, MAP and MRR than those of Lucene. Since tags contain only a few keywords, they are better suited for the ad hoc keyword search than the Lucene. However, when both subtitles and tags are combined and used as a proxy to video document, Apache Lucene achieves the best performance in retrieving the relevant videos. The BM25-based search engine (a.k.a., Lucene) delivers relevant videos for 93% of the search queries with 76% mean average precision and 0.76 mean reciprocal rank, which are highly promising. Such findings justify our choice of using Lucene as the search engine of TechTube. It also should be noted that TechTube is not restricted to Lucene and thus can be easily extended with other available search engines (e.g., YouTube search API, Indri).

Summary of RQ$_2$: Apache Lucene achieves 93% Hit@5 with 76% precision in retrieving the relevant technical videos and outperforms an ad-hoc method, which **justifies** our choice of using Lucene as the search engine.

Ad-hoc search engine deals with the only metadata files (i.e., Video Tags) better than the subtitle files, and combine tags and subtitle files. The ad-hoc search engine finds the relevant video 92% of the time, within the Top-5 retrieved videos as results with a mean average precision of 75%. The ad-hoc reciprocal rank is 0.75, which means that the ad-hoc search engine finds the most relevant video as the first or second retrieved results. The same table presents that the Lucene search engine and ad-hoc search engines retrieve the video with similar accuracy while using only subtitle files. Hence the ad-hoc technique provides 6% higher average precision and reciprocal rank. TableIII presents that the Lucene search engine improves the best setup of the ad-hoc search engine for 1% in Top-5 accuracy, 1% Mean Average Precision@5 and 0.01 Mean Reciprocal Rank@5.

D. RQ$_3$: Speaker's Silence as the Video Splitting Method

One of the major aspects of TechTube is its video splitting mechanism. The video chunking step (Section III-A1) divides each video into multiple sections, which is essential to identify the relevant video sections. We employ two video splitting methods –*speaker's silence* and *subtitle sentence*. The silence-based method splits a video into multiple sections based on the speaker's silence as explained in Section III. Following silence detection, we run Google Speech Recognition Engine [31] to convert the vocal (and non vocal) sound into the textual contents. We also split the videos using their subtitles collected from YouTube. The subtitle sentence-based method splits each video using the duration of each sentence from these video subtitles. We experiment with these two splitting methods and determine the performance of TechTube in retrieving the relevant video sections using three performance metrics – precision, recall, and F1-score. Fig. 4 summarizes our comparative analysis.

Table IV
EVALUATION OF SILENCE-BASED SEGMENTED SPEECH AGAINST THE SUBTITLE-BASED SEGMENTED SPEECH

Segmenting Technique	Metric	Mean	Median	Q1	Q3
Silence-Based	Precision	**68.39%**	0.95	0.29	1.0
	Recall	**45.48%**	0.39	0.12	0.80
	F-score	**45.21%**	0.49	0.21	0.70
Subtitle-Based	Precision	63.82%	0.99	0.11	1.0
	Recall	30.1%	0.12	0.01	0.57
	F-score	29.63%	0.19	0.02	0.58

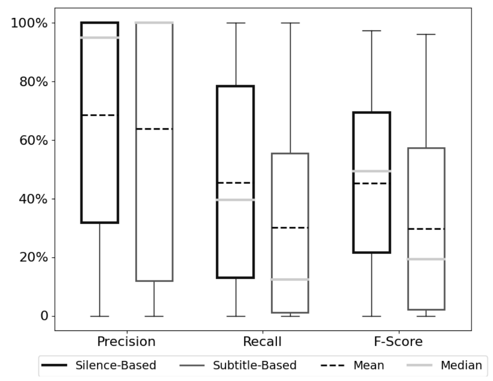

Figure 4. Comparison between silence-based video splitting and subtitle-based video splitting for the retrieval of relevant video sections

From Fig. 4, we see that the TechTube, on average, retrieves relevant fragments from the technical videos with 68% precision when silence-based splitting method is used. On the contrary, such a precision is about 64% with the subtitle-based splitting method, which is comparatively lower. However, TechTube achieves 51% higher recall and 53% higher F1-score in retrieving relevant video sections with the silence-based method than that with the subtitle-based method.

From the box plots in Fig. 4, we also notice that the mean and median measures for silence-based method are comparatively higher than those for subtitle-based method.

All these findings above justify our choice of using speaker's silence as the video splitting method of TechTube.

Summary of RQ$_3$: TechTube achieves 7% higher precision and **51%** higher recall with silence-based method than those with subtitle-based method, which **justifies** our choice of using speaker's silence as the video splitting mechanism.

Unlike prior technique, line-by-line segmenting the subtitle on average finds the relevant fragment of the videos by a mean precision of 64%, median of 0.99, recall of 30%, and an F-score of 29%. The comparison between the two mentioned segmenting techniques reports that the silence-based segmenting technique, on average, improves the mean precision, recall, and F-score for 4%, 15%, and 16%, respectively.

Table V
ROLE OF SEARCH QUERIES IN RELEVANT VIDEO RETRIEVAL

Engine	Query	Hit@1	Hit@3	Hit@5	MAP	MRR
Lucene	Baseline	59.18%	84.69%	89.79%	0.72	0.72
	Reformulated	64.28%	90.81%	**92.85%**	**0.76**	**0.76**
Ad-hoc	Baseline	54.08%	70.41%	78.57%	0.61	0.61
	Reformulated	62.24%	88.77%	**91.83%**	**0.75**	**0.75**

E. RQ$_4$: Benefits of the Query Reformulation Method

TechTube accepts free-form user queries that might not always perform well. To improve results, TechTube employs a query reformulation engine (Section III-B2) that complements these queries with important keywords from the video speeches. In particular, we select the most frequent keywords from the speeches of relevance feedback videos for query expansion (details in Section III-B2). They might have a better chance of identifying the relevant technical videos and the relevant sections from these videos. In our experiment, we investigate whether such a query reformulation improves the TechTube's performance or not. We thus experiment with two types of queries – free-form, user provided queries (a.k.a., baseline queries) and reformulated queries, and determine how our approach – TechTube – performs in retrieving the relevant videos and relevant sections from them. We use Hit@K, MAP and MRR to evaluate the retrieval of relevant videos whereas precision, recall and F-score are used to evaluate the retrieval of relevant sections from the videos. Table V and Figures 5, 6 summarize our comparative analysis.

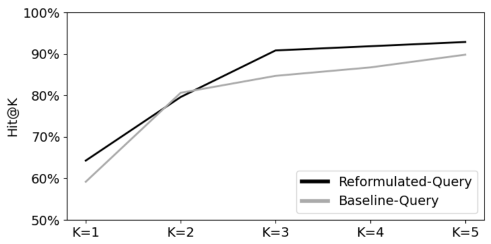

Figure 5. Comparison between baseline and reformulated search queries in the relevant video retrieval

RQ$_4$ (a)-Impact of Query Reformulation in the Retrieval of Relevant Videos: From Table V, we see that TechTube, using the baseline query, retrieves the relevant videos 90% of the time within the Top-5 results with a mean average precision of 72% and a mean reciprocal rank of 0.72. However, it performs even higher with the reformulated search query. Our approach delivers the relevant videos 93% of the time within the Top-5 results when reformulated search queries are used with Apache Lucene. It provides these relevant videos with a mean average precision of 76%, and a mean reciprocal rank of 0.76, which are 5% higher. Furthermore, the reformulated queries achieve 7% higher Hit@3 than the baseline queries in retrieving the relevant videos. We also notice that reformulated queries outperform the baseline queries when ad hoc search engine is used. From Fig. 5, we also see that the reformulated queries outperform the baseline queries for various Hit@K measures when they are executed with Apache Lucene. All these findings clearly suggest that query reformulation has a positive impact on the TechTube's performance.

Table VI
EVALUATION OF REFORMULATED QUERY AGAINST THE BASELINE QUERY FOR RELEVANT SECTION RETRIEVAL

Query Type	Metric	Mean	Median	Q1	Q3
BaseLine Query	Precision	**67.73%**	0.94	0.27	1.0
	Recall	46.42%	0.40	0.13	0.83
	F-score	46.14%	0.50	0.22	0.70
Reformulated Query	Precision	**67.46%**	0.82	0.33	1.0
	Recall	**53.44%**	0.50	0.20	0.95
	F-score	**50.47%**	0.53	0.28	0.74

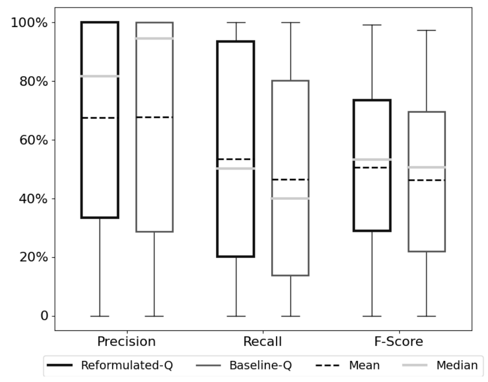

Figure 6. Comparison between baseline query and reformulated query in the retrieval of relevant video sections

RQ$_4$ (b)-Impact of Query Reformulation in the Retrieval of Relevant Video Fragments: Figure 6 shows the comparison between baseline query and reformulated query in delivering the relevant sections from the technical videos. We see that TechTube, on average, achieves 67% precision with both baseline and reformulated queries. That is, ≈70% of the retrieved video sections overlap with the ground truth. However, our approach achieves 15% higher recall and 9% higher F1-score with the reformulated query than with the baseline query.

All these findings above suggest that query reformulation mechanism improves TechTube's performance in retrieving the relevant sections from the technical videos.

Summary of RQ$_4$: Query reformulation **complements** the free-form, user provided search queries and improves TechTube's performance in the retrieval of relevant technical videos and relevant video sections from them, which **justifies** our choice of using the query reformulation.

TechTube retrieves the relevant fragment of the videos to a given baseline query with a precision of 67% compare to the *ground truth*. Furthermore, Using baseline search query in video relevant fragment retrieval, TechTube, on average, provides the retrieved video summary with a recall of 46% and an F-score of 46% as well. Besides the performance of TechTube for baseline search queries, our approach, on average, retrieves the video relevant fragment to a reformulated search query with a precision of 46%, a recall of 53%, and an F-score of 50%. The comparison between the results of the metrics for both types of search queries reports that the TechTube approach performs better while uses reformulated search queries. Despite the same precision and higher median of the retrieved fragments for the baseline search queries, reformulating the search queries using the Term Frequency method on average improves the recall of the retrieved relevant fragments for 7%, and also improves the F-score of the retrieved fragments for 4%.

F. RQ$_5$: Evaluation of TechTube with Developer Study

To evaluate the effectiveness of TechTube to assist developers in real-world development tasks, we conducted a user study. A total of 16 developers participated in the study. Each developer completed six different programming tasks using TechTube. We manually checked each completed task for accuracy. After completing the tasks, each user was invited to provide their feedback on their overall experience of using TechTube. Besides, in the second phase of the study we campare the summarized videos against the original videos using an online survey. In the remainder of the section, we briefly explain the study setting and the results.

TechTube Setup: We create a web-based prototype of TechTube that users can access from online. We download almost 500 video tutorials related to six programming tasks (Table VII) from YouTube. We then pre-process those videos using TechTube offline component.

Participants: We recruit 16 developers with more than one year of programming experience in one or more of the following languages: Python, Java, C#, Javascript, and PHP. All of these languages are among the most popular programming languages in Stack Overflow, one of the most popular Q&A sites for developers. This variety of programming languages also ensures that we can reliably determine the effectiveness of TechTube across developers of diverse expertise. Out of the 16 participants, nine were graduate students and seven were professional developers. The professional developers are found via the online site Freelancer.com. We contact all developers directly (e.g. via email and chat) and explain the whole study in details.

Table VII
PROGRAMMING TASKS SELECTED FOR THE DEVELOPER STUDY

Level	Task ID	Task Description
Easy	Task 1	Create a Binary calculator for two decimal input numbers
	Task 2	Create a program to scan QR codes
Moderate	Task 3	Create a program to find the broken links in an HTML webpage
	Task 4	Create a program to send emails
Hard	Task 5	Create a program for a client and a server to communicate over a socket
	Task 6	Create a program to measure the similarity between two texts using Cosine Similarity

Study Setup: We choose six tasks (see Table VII) based on two criteria. First, five programming languages were chosen to suit the expertise of the study participants: Python, Java, C#, PHP and Javascript. Second, The difficulty level of the tasks should vary whereas the tasks should not be tedious and should not require a long time (e.g., more than 30 minutes). In the first phase, we perform a pilot study to determine the difficulty level and the average implementation time of the tasks. Each of these tasks takes from 20 to 30 minutes to complete. To obtain comparable results, we provide each participant with a pre-configured virtual machines (VM) equipped with TechTube and the required tooling to implement their tasks (e.g., IDE). The virtual machine records all activities of a developer. Based on these VMs and their recorded videos, we identify the search queries issued by the developers for each of the tasks. Besides the queries, we also identify the video summaries generated for them and what they used as guides in the completion of the tasks. In the second phase, we aim to evaluate the relevance of the video summaries (generated by TechTube) against the developer queries issued during their tasks. We thus extract the frequently used queries for each task and the frequently retrieved video summaries for them to conduct an online survey. Since almost all of our participants have more than one-year programming experience in Python, we collect the queries related to Python programming tasks. In our survey, we provide the frequently used queries for each task (e.g., Table VIII) and the corresponding summarized videos from the TechTube and the original video from YouTube. We then ask the participants to compare the accuracy, preciseness, conciseness and usefulness of each video summary against the original video by answering several questions.

Table VIII
SEARCH QUERIES COLLECTED FROM THE DEVELOPER STUDY

Task	Search Query
Task 1	How to create a simple calculator in python?
	How to convert decimal numbers to binary numbers in python?
Task 2	How to create a QR code scanner program in python?
Task 3	How to extract the links from an Html webpage in python?
	How to send an Http request to a link and get the response in python?
Task 4	How to send emails to contact in python?
Task 5	How to create a socket and client-server communication sending and receiving data in python?
Task 6	How to convert words to the vectors in python?
	How to compare two texts with Cosine Similarity?

Study Findings: All of 16 programmers successfully performed the assigned tasks and developed them. Based on the provided screen videos, we find that each task implementation took an average of 26 minutes. We use a Likert scale of 1 to 10 to capture a participant's responses on the accuracy, preciseness, conciseness and usefulness of a video summary. Figure 7 presents the average and median of participants' responses to each question in the second phase of the study. From Figure 7, we see that, on average, participants find the summarized videos accurate and relevant to the search queries. Furthermore, with a comparison between the original videos and the summarized videos, on average, they find the video summaries to be precise and concise enough and useful for implementing their programming tasks. A follow up survey with the developers reveals that they find the video summaries to be the most useful. For example, P20 (i.e., participant number 20) commented that: *"TechTube is an amazing tool; I was able to navigate and develop the codes."* P15 also commented that: *"The summarized videos are handy because of removing the irrelevant information thoroughly for solving the problem."* The participants also find the usage of video

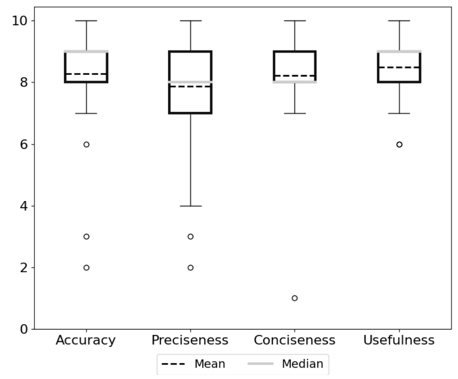

Figure 7. Developers' responses on the relevant video summaries generated by TechTube

summaries to be an easier and faster method to solve a problem. For example, P15 commented that *"watching multiple summarized videos discussing a topic is easier and faster to find the solutions rather than using the available resources on the Internet (e.g., forums, YouTube)."* . the participants also made several suggestions to extend TechTube for it to become even more useful for example, P2 suggested: *"TechTube needs to gather related Stack Overflow posts and code snippets as an enhancement."*.

Summary of RQ$_5$: The user study involving 16 developers shows that TechTube can assist developers effectively in completing diverse coding and technical tasks. Each participant was able to complete the coding tasks assigned to them using TechTube. The participants found the relevant video summaries from TechTube useful to complete their tasks.

V. THREATS TO VALIDITY

Internal validity threats relate to experimental errors. In our empirical evaluation of the retrieval of relevant videos and video segments, we compare the results against a ground truth database that we created by taking inputs from three human coders. To mitigate subjective bias, each ground truth was consulted with all three coders. Each of these human coders is a graduate student in Software Engineering. They have the necessary programming background and expertise in Python and Java to analyze the search queries and the videos. We also mitigated the bias in the user study by asking each participant to complete three programming tasks, where each participant worked remotely without the assistance or support from others. In addition, to mitigate the fatigue that may arise during the study, we asked each participant to take sufficient breaks between each task.

External validity threats relate to the generalization of the obtained results. Our approach warrants audio in a video to perform well with the natural language queries. We use speaker's silence to split the video (Section III-A1). Thus, our approach might not be applicable for online technical videos that either do not contain any audio or contain very little audio.

While the current implementation of TechTube uses technical videos from YouTube, it can use the technical videos from any online video repository. Another threat comes from our manual selection of relevant videos for the experiment. Although we carefully identify the relevant videos from a total of 400 videos, our selection might have some subjective bias due to the use of multiple people in tagging the videos sections. We have used the top StackOverflow posts titles as part of our experimental search queries where number of them might be edited by the posts authors to make them more understandable for the responders leading to having more precise result for our experiments.

VI. Related Work

Research in Software Engineering to study video tutorials is increasing with the use of videos to assist in programming or technical tasks is gaining popularity in online forums. The work of MacLeod et al. [3] is one of the first few that investigated the reason for making video tutorials by the users. By concentrating on the screencasts, they identified that the video tutorials can be complementary resources for the text-based resources (e.g., API documentations). Since the video tutorials benefit from the audiovisual trait, they are useful materials for sharing the knowledge between two developers. MacLeod et al. [3] specified this feature as the key advantage of the video tutorials rather than other written documents. Their study motivates us to perform research on the technical video tutorials. Escobar-Avila et al. [38] conduct a survey on online learning preferences involving human subjects. They report that most of their participants liked visual/auditory resources (e.g., video tutorials) and preferred multiple small videos rather than watching a long video. Thus, their work motivates our work that reduces a long video tutorial into a video summary by keeping only the essential sections.

Parra et al. [39] employ different approaches of video tagging and classification and attempt to help the developers to determine the relevance of a video tutorial in short amount of time. On the other hand, we provide the video summary from a long technical video so that the developers can determine its relevance against their query without watching the whole video. Adcock et al. [5] present an approach –TalkMiner– that relies on OCR (i.e., Optical Character Recognition) to capture keywords from the slides of a webcast. TalkMiner provides the critical slides from the webcast using a combination of OCR, text processing, and information retrieval methods (e.g., Lucene). However, the approach might not perform well with poor-quality and noisy videos due to its high dependency on the OCR technology. Thus, their work motivates us to leverage another feature of the video (e.g., speech track) in identifying the essential sections. Another study by Yadid and Yahav [40] aims to extract the source code from the video tutorial frames. They proposed an OCR-based approach mixed with statistical language models to extract the code snippets from the image frames. Despite the average precision between 80% and 82%, their work covers just the specific videos related to programming containing code snippets. Unlike our proposed approach TechTube, their work could not perform on the other type of technical video tutorials (e.g., tool installation, webinars). Most recently, Ponzanelli et al. [4] focus on the programming video tutorials. They proposed to identify the relevant fragments of the video tutorials and segment the video tutorials based on the video frames content with the corresponding audio transcript. Unlike us, they use OCR to extract the content of the video frames (e.g., code snippets). Similar to us, they also use the audio transcript beside the frame contents to classify the relevant fragments of the video. Given that their approach relies on detecting code snippets in the video tutorials, it can be limited to video tutorials that most contain code snippets. Furthermore, their algorithms and heuristics that are designed to detect the Java code might also not be suitable for other programming languages (e.g., Python). The works of Yadid and Yahav [40] and Ponzanelli et al. [4] motivate us to create TechTube. Unlike the above techniques that are only applicable to video tutorials containing code snippets, TechTube is suitable for any technical videos that may or may not contain code snippets. TechTube thus complements the state of art research in software/technical video processing and segmentation to assist in diverse tasks related to software engineering.

VII. Conclusion And Future Work

In this research, we propose a novel approach – TechTube – that identifies the relevant sections of a technical video tutorial to a search query and delivers them as a summarized coherent, video fragment. TechTube gives developers the capability of using natural language queries to retrieve the relevant sections. To create a precise and concise fragment, TechTube reformulates the query and matches it with technical videos available in an online repository. TechTube uses a silence-based method to better identify the relevant sections of a technical video tutorial. Our evaluation of TechTube against 98 user-generated search queries shows that TechTube can retrieve the relevant video tutorials in 93% of the time. TechTube can also properly highlight the relevant video segments in a video with a precision of 67% and a recall of 53%. We compared TechTube with a state of the art approach from the literature –CodeTube– and found that TechTube outperforms this approach significantly. We conducted a user study involving 16 developers to evaluate the effectiveness of TechTube during the completion of six development tasks. All study participants were able to complete the development tasks using TechTube. They also find the summarized videos precise and concise enough to use them in implementing programming tasks. Our future work will focus on extending TechTube to check for the efficiency of different query retrieval techniques within the TechTube search engine module and to study how TechTube can complement traditional text-based software documentation resources.

References

[1] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two Studies of Opportunistic Program-

ming: Interleaving Web Foraging, Learning, and Writing Code," in *Proc. SIGCHI*, 2009, pp. 1589–1598.

[2] J. Burgess, "Youtube," in *Oxford Bibliographies Online*, L. H. Meyer, Ed. United Kingdom: Oxford University Press, 2011, pp. 1–1. [Online]. Available: https://eprints.qut.edu.au/46719/

[3] L. MacLeod, M. Storey, and A. Bergen, "Code, camera, action: How software developers document and share program knowledge using youtube," in *2015 IEEE 23rd International Conference on Program Comprehension*, 2015, pp. 104–114.

[4] 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," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 261–272. [Online]. Available: https://doi.org/10.1145/2884781.2884824

[5] J. Adcock, M. Cooper, L. Denoue, H. Pirsiavash, and L. Rowe, "Talkminer: A lecture webcast search engine," 10 2010, pp. 241–250.

[6] M. Alahmadi, J. Hassel, B. Parajuli, S. Haiduc, and P. Kumar, "Accurately predicting the location of code fragments in programming video tutorials using deep learning," in *Proc. PROMISE*, 2018, p. 2–11.

[7] J. J. Rocchio, *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc.

[8] M. M. Rahman, C. K. Roy, and D. Lo, "Automatic query reformulation for code search using crowdsourced knowledge," *Empirical Software Engineering*, vol. 24, no. 4, pp. 1869–1924, Aug. 2019.

[9] TechTube. (2020, May) Techtube replication package. [Online]. Available: https://github.com/TechTube/TechTube-Replication-Package

[10] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel, "Strategies for automatic segmentation of audio data," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, vol. 3, 2000, pp. 1423–1426 vol.3.

[11] T. Theodorou, I. Mporas, and N. Fakotakis, "An overview of automatic audio segmentation," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 6, no. 11, p. 1, 2014.

[12] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, vol. 1, 2000, pp. 452–455 vol.1.

[13] S. K. Bajracharya and C. V. Lopes, "Analyzing and mining a code search engine usage log," *Empirical Softw. Engg.*, vol. 17, no. 4–5, p. 424–466, Aug. 2012. [Online]. Available: https://doi.org/10.1007/s10664-010-9144-6

[14] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The vocabulary problem in human-system communication," *Commun. ACM*, vol. 30, no. 11, p. 964–971, Nov. 1987. [Online]. Available: https://doi.org/10.1145/32206.32212

[15] D. Liu, A. Marcus, D. Poshyvanyk, and V. Rajlich, "Feature location via information retrieval based filtering of a single scenario execution trace," in *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 234–243. [Online]. Available: https://doi.org/10.1145/1321631.1321667

[16] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1589–1598. [Online]. Available: https://doi.org/10.1145/1518701.1518944

[17] C. Sadowski, K. T. Stolee, and S. Elbaum, "How developers search for code: A case study," in *Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 1600 Amphitheatre Parkway, 2015.

[18] K. Kevic and T. Fritz, "Automatic search term identification for change tasks," in *Companion Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE Companion 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 468–471. [Online]. Available: https://doi.org/10.1145/2591062.2591117

[19] M. M. Rahman and C. Roy, "Effective reformulation of query for code search using crowdsourced knowledge and extra-large data analytics," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 473–484.

[20] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *ACM Comput. Surv.*, vol. 44, no. 1, Jan. 2012. [Online]. Available: https://doi.org/10.1145/2071389.2071390

[21] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. USA: Cambridge University Press, 2008.

[22] M. M. Rahman and C. K. Roy, "Improved query reformulation for concept location using coderank and document structures," in *Proc. ASE*, 2017, pp. 428–439.

[23] W. H. Gomaa, A. A. Fahmy *et al.*, "A survey of text similarity approaches," *International Journal of Computer Applications*, vol. 68, no. 13, pp. 13–18, 2013.

[24] M. Vijaymeena and K. Kavitha, "A survey on similarity measures in text mining," *Machine Learning and Applications: An International Journal*, vol. 3, no. 2, pp. 19–28, 2016.

[25] T. Liu and J. Guo, "Text similarity computing based on

standard deviation," in *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 456–464.

[26] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in *2016 4th International Conference on Cyber and IT Service Management*, 2016, pp. 1–6.

[27] Z. Shi, J. Keung, and Q. Song, "An Empirical Study of BM25 and BM25F Based Feature Location Techniques," in *Proc. InnoSWDev*, 2014, pp. 106–114.

[28] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. Cambridge Uni Press, 2009.

[29] FFMPEG. (2000) A complete, cross-platform solution to record, convert and stream audio and video. [Online]. Available: https://www.ffmpeg.org/

[30] J. Robert. (2018) Pydub: Manipulate audio with a simple and easy high level interface. [Online]. Available: http://pydub.com/

[31] A. Zhang. (2014, Apr.) Speech recognition. [Online]. Available: https://pypi.org/project/SpeechRecognition/

[32] G. Salton and C. Buckley, "Readings in information retrieval," 1997, ch. Improving Retrieval Performance by Relevance Feedback, pp. 355–364.

[33] S. Haiduc, G. Bavota, A. Marcus, R. Oliveto, A. De Lucia, and T. Menzies, "Automatic Query Reformulations for Text Retrieval in Software Engineering," in *Proc. ICSE*, 2013, pp. 842–851.

[34] G. Gay, S. Haiduc, A. Marcus, and T. Menzies, "On the Use of Relevance Feedback in IR-based Concept Location," in *Proc. ICSM*, 2009, pp. 351–360.

[35] P. Sojka and M. Líška, "Indexing and searching mathematics in digital libraries," in *International Conference on Intelligent Computer Mathematics*. Springer, 2011, pp. 228–243.

[36] M. M. Rahman and C. Roy, "Nlp2api: Query reformulation for code search using crowdsourced knowledge and extra-large data analytics," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 714–714.

[37] M. Paterson and V. Dancík, "Longest common subsequences," 1994, p. 127–142.

[38] J. Escobar-Avila, D. Venuti, M. Di Penta, and S. Haiduc, "A survey on online learning preferences for computer science and programming," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2019, pp. 170–181.

[39] E. Parra, J. Escobar-Avila, and S. Haiduc, "Automatic tag recommendation for software development video tutorials," in *Proceedings of the 26th Conference on Program Comprehension*, ser. ICPC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 222–232. [Online]. Available: https://doi.org/10.1145/3196321.3196351

[40] S. Yadid and E. Yahav, "Extracting code from programming tutorial videos," in *Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, ser. Onward! 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 98–111. [Online]. Available: https://doi.org/10.1145/2986012.2986021