Mapping educational software engineering content on YouTube

Maša Pejović*
Nikola Vasović*
masa.pejovic@unimediteran.net
nikolav706@gmail.com
Mediterranean University
Podgorica, Montenegro
Mälardalen University
Västerås, Sweden

Antonio Cicchetti
antonio.cicchetti@mdu.se
Mälardalen University
Västerås, Sweden

Robbert Jongeling
robbert.jongeling@york.ac.uk
University of York
York, United Kingdom
Mälardalen University
Västerås, Sweden

ABSTRACT

Online educational videos offer an accessible means for persons without formal training to learn software engineering concepts beyond programming alone. However, it is difficult to get an overview of what software engineering training content is available, which is a prerequisite for more detailed analysis such as on how that content influences the perspectives of these citizen developers. In this paper, we report on our attempt to systematically map the offer of such videos on YouTube. Our contributions are twofold: (1) We design and implement a pipeline for the automated retrieval and relevance scoring of YouTube videos; and (2) we use this pipeline to perform two mappings of software engineering training videos. We identify considerable challenges in performing this type of study and provide advice for future research using this methodology.

CCS CONCEPTS

• **Social and professional topics** → **Informal education**; • **Software and its engineering** → *Software creation and management*.

KEYWORDS

Software engineering education, Citizen developers, Mapping study, Online video

ACM Reference Format:
Maša Pejović, Nikola Vasović, Antonio Cicchetti, and Robbert Jongeling. 2026. Mapping educational software engineering content on YouTube. In . ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As a software engineering community, we increasingly expect citizens to be more self-reliant in their involvement with software engineering activities such as requirements elicitation [14] and even software development as citizen developers [1]. This reliance can be expected to only increase in the future under the influence of large language models (LLMs). Tools such as Lovable[1] are aimed towards supporting software development by such persons without formal software engineering training. For learning programming, there is plenty of material available online, such as in Q&A websites, blogs, LLM-powered chat-bots, and video tutorials. There is, however, a lot more to software engineering than just programming. For example, software engineers must incorporate in their work ethical concerns, regulations surrounding safety and cybersecurity, and much more.

We are interested to learn what type of software engineering training is available for those members of the public who have not received a formal software engineering education. Therefore, we look for training material available in informal form, specifically YouTube (YT) videos and not for other forms such as massive open online courses (MOOCs) on other platforms. YT is the biggest online video platform hosting billions of videos and thus it has the potential to serve as a resource for self learning [26]. Our premise is that among all those videos, there must be suitable training material for learning software engineering topics. Nevertheless, it is not so easy to get an overview of what content there is for citizen developers. Therefore, we perform a mapping study of YT videos, adapting the traditional approaches for mapping the scientific literature.

The goal of this research is to map educational software engineering content on YT. This is a vast search space and therefore the secondary goal of this research is to provide automated means to perform such a mapping. We report on our process of doing this research, which includes developing a method on how to do such a mapping study in the first place, and show how that has also impacted the scope of what we could research. Throughout the research, we narrowed our scope guided by feasibility of doing the mapping, starting from software engineering as a whole and later focusing on specific aspects within that.

We define the following two research goals:

RG1. To develop automated support for mapping studies of YT.
RG2. To apply the resulting tooling from RG1 to discover trends in videos about software engineering topics on YT.

As a part of trying to understand how to do such a mapping study, we need to understand the type of video that citizen developers would be looking for and the way that they would look for these videos (the latter is relevant because uploaders change their video titles to match either search habits or clicking habits of viewers). It is not obvious what search parameters need to be put in place to find videos relevant to viewers. Therefore, as part of this study, we conduct a questionnaire survey (see Section 3.2) among citizen developers to elicit insights into how they use YT for their own learning. We use the answers from the 52 respondents in the further design of our mapping study.

*Both authors contributed equally to this research.
[1]https://lovable.dev/

Conference'17, July 2017, Washington, DC, USA
2026. ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/nnnnnnn.nnnnnnn

This paper brings contributions in two aspects. First, we develop a pipeline and guidelines for doing automated mapping studies on YT. This answers many non-trivial open questions about doing such work. The complete implemented pipeline is provided as supplementary material to the paper [29]. Second, we show that this pipeline can indeed result in an overview of the available videos on a topic and show a heatmap of most popular software engineering topics throughout the latest years. These mappings show an interest in software maintenance as a video topic. Both aspects together show the potential for this research method as a way to research available software engineering training videos.

The remainder of this paper is structured as follows. We relate our work to the literature in Section 2. We list requirements for our mapping study and apriori challenges to overcome in Section 3. We detail our method, its implementation and validation in Section 4. We present results in Section 5 and discuss further encountered challenges, limitations, and threats to validity in Section 6. We conclude and propose future work in Section 7.

2 RELATED WORK

In this section, we discuss previous mapping studies of YT and previous studies on software engineering training videos.

Reported sources of data other than the video itself to judge YouTube videos include number of views and likes, comments, and metadata [12, 28]. The possibly available metadata is extensive and includes also view patterns of the audience. It has been shown that, in general, longer-watched videos also attract more likes per view and more comments, thus view duration is logically a good proxy for the relevance of the video to the audience [19]. However, what is missing in this perspective is an indication of the relevance of the content of the video for a certain topic, which is what we focus on in this paper.

Researchers from other domains than software engineering have performed various types of mappings of YT. One study gathered metadata from more than 36 million channels to chart genre hierarchies and national clusters across the platform as a whole [22]. This study was exclusively quantitative, based on metadata of videos and did not include a judgment on relevance of the videos. Another study considered 317 videos on sleep apnea to understand what information patients are exposed to [31]. The study showed that certain metadata such as video duration and number of views can be used as a proxy for video quality.

When comparing software engineering curricula and the needs of employers, researchers found a good correspondence between what is taught and what is needed in the area of coding and design, but a lack of teaching in project management, configuration management, and other practical process areas [11]. Our starting point is that we expect that professionals as well as citizen developers search for more knowledge on these topics from alternative sources, and that one of these sources may be YT videos. In part due to the COVID-19 induced move to online classes, lecturers have published YT tutorials to replace in-person demonstrations [10]. Indeed, YT is reported as a source of learning material for both formal and informal learning [27]. Despite the popularity of the medium still, a common concern is the quality of the content [27].

3 REQUIREMENTS FOR A MAPPING STUDY OF YOUTUBE

In this paper, we consider YouTube videos on software engineering topics beyond programming and are interested to find out what SE topics receive the most attention and what areas are underrepresented. We therefore design and perform a mapping study to identify trends in published videos.

3.1 The need for automation

The volume of YT (20 billion videos as of April 2025 [5]) prevents a manual search and comprehensive mapping of all content related to such a broad topic as software engineering. Moreover, given the frequency with which new videos appear (recent estimates state 20 million videos are uploaded daily [5]), such an analysis would risk to be outdated before it is finished, much more so than in traditional mapping studies of the scientific or even gray literature. Manually retrieving metadata, transcripts, and engagement statistics for every candidate video is time-consuming and practically impossible due to this scale. Therefore, we develop an automated pipeline that retrieves this information, applies consistent filters, and periodically refreshes the dataset, aiming for reproducibility and up-to-date information as much as possible in such a dynamic setting.

A core challenge of automating the mapping is judging the relevance of videos. It is not about number of views or likes, it is rather about: is this video a good fit for the topic we are trying to map?

In our mapping study, we adapt the methodological guidelines for systematic mapping studies provided by Petersen et al., and consider instead YT videos as the equivalent of "primary studies" from the literature [20]. Table 1 lists our adaptations, and they are further elaborated in Section 4.

3.2 Relevant topics and search strategies – a survey among citizen developers

Searching YT requires a different focus than searching the literature, making the usual experiences of secondary studies not always applicable. For example, a number of results for a given search string is not readily available in searching YT, due to the imprecision of the "*totalResults*" parameter that the API reports (more in Section 4). Searching is also perilous since there is not a similar predictability of terms used in video titles as there is for academic papers. Moreover, there is limited insight into what people who would be interested in these topics would actually search for. Therefore, we have conducted a questionnaire survey among persons self-identifying as citizen developers to find out (1) topics of interest to citizen developers, and (2) search strategies of citizen developers. The complete results of the survey are available as supplementary material [29]. We used the insights from this survey as input to narrow down feasible and relevant topics for a mapping study.

To save space in this paper, we limit the reporting about this questionnaire survey to its main results as follows. We spread a questionnaire survey through student networks to explain the intent of the research and to solicit responses. Eventually, we got 52 responses from persons self-identifying as citizen developer, which we verified using the first two questions in which we asked (i) what software they had built or maintained in the past year (they should have), and (ii) whether they had received any formal SE education

Table 1: Contrasting aspects of typical Systematic Mapping Studies with this paper.

Aspect	Typical SMS [16]	Our mapping study of YouTube
Corpus / search space	Digital libraries such as Google Scholar, Scopus, ACM digital library, and IEEE Explore	*Single* venue: YouTube (accessed through its Data API v3)
Unit of analysis	Peer-reviewed publication	Individual video (with auto-generated transcript)
Topic identification	Manual coding / keyword tagging by researchers	LDA topic-modeling + entropy-weighted TF–IDF, topics mapped to SWEBOK [30] areas
Classification facets	Topic, contribution type, research method, publication venue, publication year	SWEBOK [30] knowledge area, publication date
Data extraction	Metadata + researcher-assigned codes	API metadata + mined transcript text (no human coding)

or training (they should not have). We asked respondents to express their interest in software engineering topics selected from Software Engineering Body of Knowledge (SWEBOK) [30] chapter titles. The three topics of most interest were "Software Requirements", "Software Maintenance", and "Software Quality". Conversely, the three topics of least interest were "Software Engineering Operations", "Software Architecture", and "Software Management". The respondents reported various search strategies to find videos corresponding to their interest area.

The respondents were further asked about their criteria for choosing to watch certain videos over others. The most often mentioned video selection criterion was its relevance to the person's current project (34/52 responses, 65%). Other factors were mentioned fewer times: the presenter's experience (26/52, 50%), the video's popularity (26/52, 50%), and the video length (16/52, 31%).

Lastly, we asked the respondents an open question about their search string: "How would you look on YouTube for a tutorial on writing a formal requirements document?" We found that the search strings people use (1) commonly include concrete SE terms such as "requirements document"; (2) typically begin with indicators about the type of instruction content they are looking for, such as "how to", or "tutorial on"; and (3) typically include audience qualifiers, such as "for beginners", or format qualifiers such as "template" or "guidelines".

The input from the questionnaire gives us insight into topics that citizen developers would find interesting and the way they would search for corresponding training videos. It does not yet help us to decide a feasible level of granularity of a mapping study, which we continue to experiment with in Section 4.

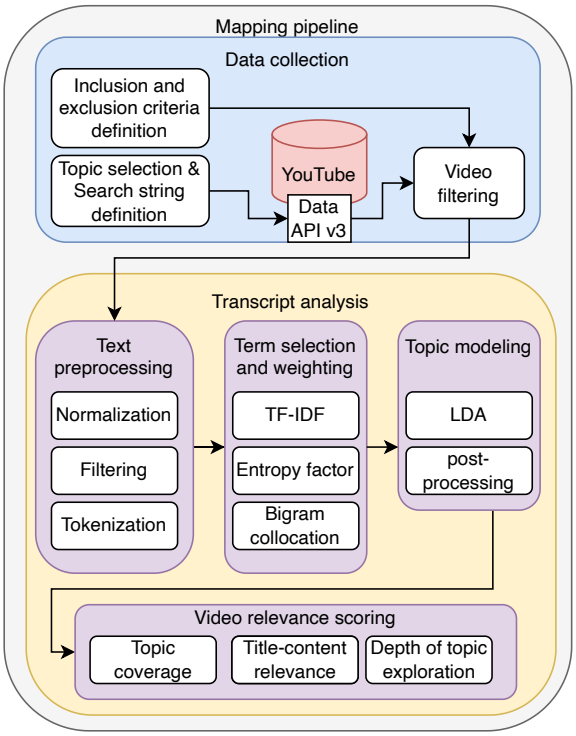

Figure 1: Overview of the research pipeline followed for automated mapping of SE videos on YouTube, the steps after the initial two are automated.

4 METHOD

In this section, we first provide a high-level overview of our mapping study, and then detail each step. Our starting point was that we wanted to create a repeatable and automated means of mapping content on YouTube to be able to manage the scale of the platform. Such an approach should be able to collect videos, assess their relevance, and store metadata for further study. We used the newest available version of the YouTube Data API v3 [13] as the primary data entry point, enabling programmatic access to video metadata and perform several steps to automatically process the transcripts. Figure 1 shows an overview of the followed steps for this work, which are illustrated in details in the following subsections.

4.1 Data collection

We used the YT Data API v3 [13] to collect videos' metadata and stored the collected data, including titles, descriptions and publication dates in a structured JSON format for further analysis.

Following the methodology of a mapping study, we formulated the following *inclusion criteria*:

I1. Video shall be relevant to the selected topic (one of the SWEBOK topics: *Software Requirements*, *Software Design*, *Software Testing*, *Software Maintenance* and *Software Quality*).
I2. Video shall have English language audio.

I3. Video shall be published between January 1st 2015 and May 1st 2025.

And these *exclusion criteria*:

E1. Video has duplicates in the collected dataset.
E2. Video is not publicly available (e.g., YT premium).
E3. Video is flagged for YT policy violations.
E4. Video lacks an automated transcription.

The search is automated using a script that encodes the inclusion criteria and queries the API. Videos that met any of the exclusion criteria or did not meet one or more inclusion criteria were discarded.

4.1.1 Topics and search strings. A study analyzing the transcripts of 130 software engineering vlogs showed the potential of using this data as a means to categorize videos [6]. One challenge to overcome is the skew of the YT recommendation algorithm, which is known to be biased, e.g. towards pseudo-scientific content [18]. Following the personalized "watch-next" suggestions also shows how conspiracy theories show up in a person's recommendations [9]. Indeed, a study using persona-based crawling showed the emergence of content "bubbles" of one-sided political content [17]. We therefore avoid relying on any personalized recommendations in our study and query the YT Data API without cookies nor personal account.

A search in the API takes as input a search string. To find our search string for the broad domain of software engineering, we consider the knowledge areas defined by the SWEBOK [30]. We have performed a unique search for each individual knowledge area. Our mapping study is then the union of these searches. To limit the scope and required number of API calls, we selected five topics from SWEBOK that were most of interest for the respondents to our questionnaire (see Section 3.2). The search strings for each sup-topic are taken to be identical to the chapter names in SWEBOK.

4.1.2 Video filtering. Videos were collected through paginated API requests. Each page contains up to 50 videos and we have chosen to limit our collection to at most 10 pages per topic, if that many existed. We have used the default order setting, by which the API returns results in order of relevance to the search query [13]. Collecting at most 500 videos per topic is a limitation we accept in our mapping due to the limited availability of API tokens. Moreover, we do not aim for an exhaustive dataset of videos on a topic, but rather try to map the way topics are covered in the most relevant videos. In any such study, there is a trade-off to be made considering the amount of data that can be collected, in this case also limited by the number of available API tokens.

4.2 Transcript analysis

The second stage of the pipeline is to analyze the videos transcripts with the goal of assessing their relevance to the topic. YT offers automatic transcripts for English-language videos, but their availability and reliability depends on audio quality and uploaders are able to disable the feature for their videos.

The pipeline for processing the textual transcripts includes four steps, with the goal of extracting only terms useful for judging the video's relevance. First, we pre-process and normalize the text to remove noise and ensure uniformity (Section 4.2.1). Second, we extract domain-specific terms, which highlight the technical vocabulary necessary for accurate analysis (Section 4.2.2). Third, we apply topic modeling using Latent Dirichlet Allocation (LDA) to uncover themes within the text (Section 4.2.3). Fourth, the topic model is refined through topic coherence enhancement (Section 4.2.4). We now detail the implementation of each step.

4.2.1 Text preprocessing. The text processing stage consists of three steps: *normalization*, *filtering*, and *tokenization*.

The transcripts generated by YT contained timestamps and speaker labels that we can remove to obtain a cleaner starting point for the analysis. Therefore, we begin with *normalization* of the transcripts by stripping timestamps, punctuation, numbers, and any characters between parenthesis, and converting all characters to lowercase.

We further filter out platform-specific filler words such as YT calls to action (e.g., "*like and subscribe*"). Prior work has shown that building a corpus specific stop-list significantly improves model interpretability: for example, one study demonstrated that removing empty and domain irrelevant words before and after training can enhance topic coherence without losing relevant content [24], and benchmarks of filler word detection have emphasized the importance of filtering non informative speech phrases to sharpen analyses [32]. Table "*stopwords*" in the supplementary material [29] features of all words and phrases we filter out from the video transcripts we collect. This filtering removes language that is characteristic of the YouTube platform rather than the software-engineering domain, preserving the accuracy of subsequent weighting and topic-modeling stages.

Next, we prepare the input for *term frequency–inverse document frequency* (TF–IDF) and topic modeling of the transcript using the `word_tokenize` function from NLTK [2]. For each token we generate *n-grams*, continuous sequences of *n* words. *Unigrams* (single words) capture the basic vocabulary, while *bigrams* (two-word sequences) such as "version control" often encode concepts that cannot be inferred from their parts alone. Not every bigram is meaningful. We measure the strength of association between its two words using Pointwise Mutual Information (PMI) [7]. PMI compares the observed joint frequency with the frequency expected if the words were independent, a high PMI value signals a collocation worthy of further analysis. The final output of this pre-processing stage is thus a cleaned list of unigrams and high-PMI bigrams that can now be ranked for specificity and weighting.

4.2.2 Term selection and weighting. After pre-processing, the remaining unigrams and bigrams need to be assigned a weight to determine their relevance to the topic. To answer this, we rank tokens with a two–stage weight that combines TF-IDF and an *entropy* bonus.

TF–IDF assigns a high score to words that occur often in one document but rarely across the whole collection [23]. Although TF-IDF already rewards distinctiveness, it does not distinguish based on frequency of occurrence across multiple documents. We therefore add an entropy factor [21, 25]: the Shannon entropy of a term's document–frequency distribution. Low entropy means the word is concentrated in a small subset of videos and is therefore more likely to mark a specific subtopic. Our proposed weighting procedure is:

(1) Compute the standard TF–IDF matrix.
(2) Calculate Shannon entropy for every term.
(3) If a term's entropy falls below a tuned threshold, multiply its TF–IDF by 1.5 to obtain the final importance score.

We selected this 1.5 boost after experimenting with a range of factors between 1.2 and 2. In particular, a balance is needed: too much boost causes a dip in term diversity, while too little boost yields more noise. We found that 1.5 yields the best recall of niche terminology; the relevant terms are clearly visible to the final output and irrelevant terms, when still present after filtering, are suppressed.

The remainder of our designed procedure is detailed as follows. Single words alone might miss many technical expressions that are only meaningful as fixed phrases, e.g., *version control* and *test harness*. To retrieve these we propose to run a bigram collocation module:

(1) A bigram finder enumerates adjacent non-stopword tokens that survive pre-processing and appear at least five times.
(2) Each candidate bigram is scored with Pointwise Mutual Information (PMI), which measures how much more often the pair co-occurs than chance would predict [7].
(3) Bigrams that exceed a PMI threshold and pass the domain-specificity filter are added to the vocabulary and weighted with the same TF–IDF + entropy scheme.

Including statistically significant collocations ensures that multi-word software engineering concepts enter the analysis with appropriate weight, preventing their component words from being treated as independent and diluting the transcript analysis.

4.2.3 Topic modeling implementation. Topic modeling clusters co-occurring terms and assigns each document a proportion over those clusters [3]. We can use this technique to discover themes from our weighed terms. We choose Latent Dirichlet Allocation (LDA), a generative Bayesian model that represents every transcript as a mixture of K topics and every topic as a probability distribution over the vocabulary [4]. LDA is well suited here because it scales to thousands of moderately sized documents, produces human-interpretable term lists, and is implemented efficiently in the Python library `scikit-learn`.

Raw LDA output often contains duplicate stems or overly general words. We therefore apply three cleaning passes: (i) stem matching merges near-duplicates; (ii) bigrams are preferred over unigrams when both appear in the same topic list, because phrases such as *version control* convey more precise meaning; (iii) any remaining generic tokens are removed by cross-checking against the stop list. For every surviving term we record its corpus-wide frequency and its normalized topic weight so results can be compared across topics.

4.2.4 Video-level relevance scoring. The topic model summarizes vocabulary, but does not yet judge the relevance of videos for the searched topic. To do so, we need to distinguish videos by the depth of coverage of topics. We developed a scoring scale for each video on three criteria: (i) topic coverage (0–4 points), (ii) title–content relevance (0–3 points), and (iii) depth of topic exploration (0–3 points).

To evaluate the consistency between a video's metadata and its spoken content, the relevance scoring algorithm concatenates the title, description and tags, then performs three independent overlap checks. It calculates the proportion of relevant title words (words longer than three characters) that also occur in the transcript. It verifies whether the explicit SWEBOK topic name appears in the transcript and it measures how many of the topic's key phrases can be found in the combined metadata.

The depth component measures how thoroughly a video treats its topic by combining observable features of the transcript. First, the algorithm derives a technical term density score by computing what fraction of the unique words drawn from the topic's key phrase list also appear in the transcript. Second, it awards up to 0.5 point for sheer length, scaling the score linearly until the first one thousand words have been reached, beyond which additional text no longer increases the total. Third, it adds 0.5 point when it detects discourse markers that signal an organized structure (for example, phrases such as "first", "finally" or concluding statements like "in summary"). A further 0.5 point is granted if the transcript contains phrases that introduce practical illustrations, such as "for example" or "case study", assuming these indicate applied rather than purely abstract treatment. Finally, another 0.5 point is available when technical cue words such as "algorithm", "architecture" or "framework" are present, reflecting a deeper, more specialist discussion.

4.3 Validation of the relevance scoring

To validate the automated relevance scoring system, we compared its output with a manually graded sample of videos, following the same rating guidelines as described in Section 4.2.4. Two independent human raters judged a stratified random sample of 30 videos from one of our collected datasets, 10 videos from 3 randomly chosen SE topics, used in the data collection phase.

We pre-determined guidelines for giving certain scores on these scales. These guidelines are available in supplementary material [29]. After viewing each of the 30 videos, raters assigned three separate ratings for the three criteria. The complete tables of ratings are provided in supplementary material, too.

To ensure the reliability of the human scores, we applied the quadratic-weighted Cohen's κ coefficient to the first round of ratings. According to the conventional Landis–Koch interpretation, the first value indicates substantial agreement ($\kappa \geq 0.70$), while the second and third fall into the moderate-agreement band ($0.41 \leq \kappa < 0.70$). Inter-rater reliability exceeded the generally accepted threshold of $\kappa \geq 0.70$ in all categories—main coverage ($\kappa = 0.708$), title–content match ($\kappa = 0.730$), and depth of topic exploration ($\kappa = 0.739$). For each video, the two human raters' scores were first summed across the three categories and then averaged, yielding a single composite score. Cohen's κ was subsequently recomputed between these averaged human scores and the predictions generated by the proposed tool, resulting in $\kappa = 0.642$. This value indicates moderate agreement between the humans and our system and therefore we conclude that our automated pipeline is moderately suitable for assessing relevance according to our criteria. Human raters typically disagreed with the tool in cases of videos that are advertising or are shorts, that score high due to matching the predefined phrases and words, but humans find irrelevant (and rightfully so).

5 RESULTS

We demonstrate our automated pipeline by performing mappings at two levels to get a better insight into what a good level of granularity would be for a mapping study of YT. This is needed since, as opposed to literature reviews, there is a lack of reference studies and researcher experience that would indicate an appropriate level of granularity of a mapping study. Moreover, the two mappings show two distinct capabilities of the pipeline, providing a broad overview of the field and distinguishing more nuanced topics. Lastly, the collection of two datasets is a compromise driven by the practical limitations explained in Section 4. A single comprehensive dataset could simultaneously show the distribution of all SWEBOK areas without loosing data about subtopics present within them, but such a dataset is infeasible to create taking into account the limitations of the Data API.

We now describe the results from the two mapping studies. The first is a "high-level" mapping in which we took five different SWEBOK topics and used them as search strings. The second is a "low-level" mapping and was made by taking the most popular topic from the first dataset and searching for their sup-topics in SWEBOK. We detail the quantitative distributions of videos across years and topics, the term patterns extracted from transcripts, and the quality scores assigned to individual videos. The results provide insights into what kind of software engineering learning material is available for citizen developers on YT.

5.1 High-level dataset overview

For the high-level dataset we collected 2500 videos, 500 videos on each of the five SWEBOK areas of *Software Design*, *Software Maintenance*, *Software Quality*, *Software Requirements* and *Software Testing*.

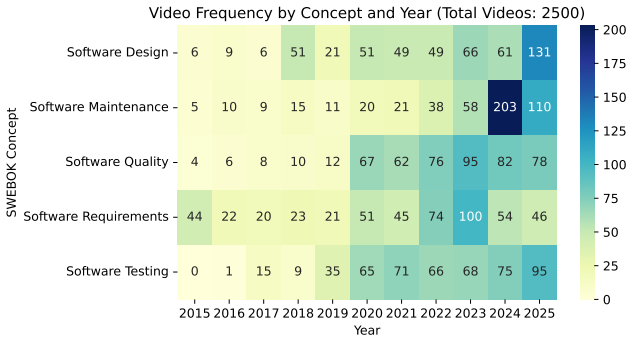

Figure 2: Heatmap of "high level" set

Figure 2 shows the frequency of uploaded videos by year. The total number of yearly uploads across all topics is the lowest in 2016 with 48 videos (2% of the dataset) and the highest in 2024 with 475 videos (19% of the dataset). We see a trend in upload frequency, increasing since 2020. The total number of videos published from 2015-2019 is 373 (15% of the dataset), the most recent six years account for 2127 videos (85% of the dataset). More than half (54%, 1322 videos) of the dataset represents uploads between 2023 and 2025.

5.2 High-level transcription analysis results

All 2500 videos in the high-level dataset were analyzed using the transcript analysis pipeline detailed in Figure 1. The result is an overview of relevant key terms collected from the videos, an overview is shown in Figure 3. All these 20 most relevant terms achieved an automatically calculated relevance score of 100%. We observe technically relevant terminology and an apparent interest in requirements, with it being the most frequently occurring term, and several related terms occurring in the top 20 too.

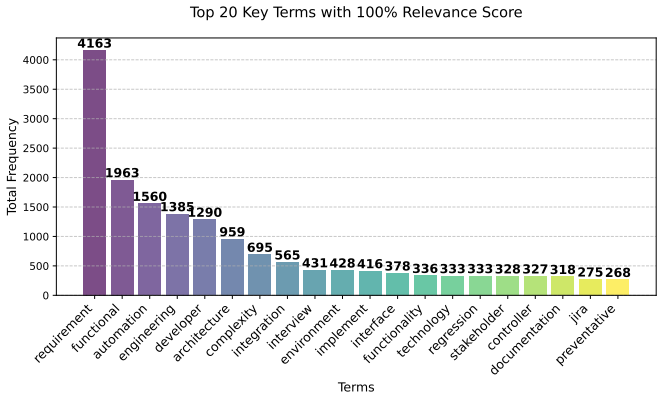

Figure 3: Bar chart of the most frequent key terms detected from the dataset

The five most common terms together represent 62% of the total, while the bottom ten terms represent only 20%. This could indicate that the videos stay largely on-topic. Essentially, the filtered transcripts emphasize a handful of key concepts and stray from them relatively little, providing a focused coverage.

5.3 Low-level dataset

We collected the low-level dataset with an identical pipeline configuration, including the same YouTube Data API endpoint, English language filter, and 2015-2025 date range. Moreover, we repeated the full pre-processing, term-weighting and topic-modeling steps. The only adjustment lies in the five narrower search strings, again taken from the SWEBOK, but now as subtopics under Requirements Engineering: "*Functional Requirements*", "*Requirements Analysis*", "*Requirements Elicitation*", "*Requirements Management*" and "*Requirements Specification*". These more narrow search strings yield fewer search results too: in this second mapping, we collected 275 distinct videos. The resulting imbalance across topics is therefore a consequence of the available content as returned as results by the API for each corresponding search string, and unrelated to the implementation details of our pipeline. We show the frequency of videos by topic in the low-level dataset in Figure 4.

Figure 4 shows that *Requirements Management* is the largest category with 85 items (31% of the dataset). Similar to the high-level dataset, the trend shows a higher number of videos in the dataset from more recent years. The year 2023 has the most uploads, 73 videos (27% of the dataset). Together, 2022 through 2024 supply

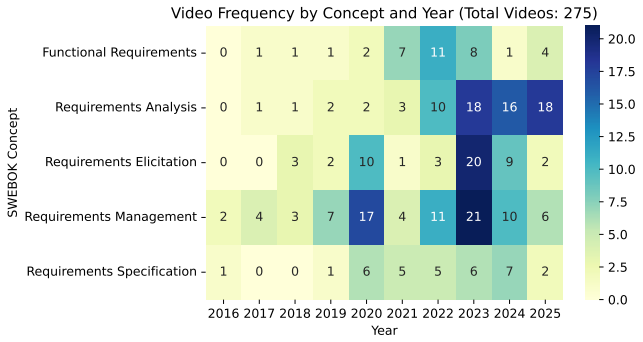

Figure 4: Heatmap of "low-level" set

56% of the dataset. As in the high-level dataset, earlier years contribute fewer videos. The distribution of results is thus uneven, with both concept totals and yearly totals skewed toward *Requirements Management* and toward the most recent three-year period.

One observation from these results is an apparent interest in process-oriented material. Videos on *Requirements Management* alone supply almost one-third of the corpus, roughly double the number devoted to *Functional Requirements* or *Requirements Specification*. This might suggest that YouTube creators emphasize how to organize and track requirements more than how to write or structure them.

5.4 Low-level transcription analysis results

We performed the same transcript as before on the low-level dataset of 275 videos. The outcomes of this run are reported in Figure 5.

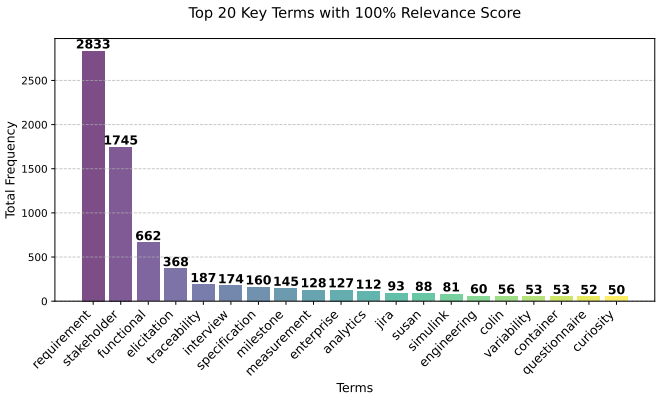

Figure 5: Bar chart of the most frequent key terms detected from the dataset

The low-level requirements corpus contains 7227 appearances of the top twenty most frequent terms which achieved a 100 % relevance score. We see as most frequently occurring keywords those that are relevant to the discussed requirements engineering topics, such as *Stakeholder*, *Functional*, *Elicitation*, and *Traceability*. These five words together account for 80% of the frequent terms.

The data implies, similarly to our findings in the high-level dataset, that videos stick to a few topics at a time. This data also suggests that most presenters concentrate on defining what the system must do and on identifying who owns those needs. Other concepts, such as interview techniques and documentation nuances, appear to be discussed less frequently.

6 DISCUSSION

In this section, we discuss challenges encountered during the mapping study, which are an important part of the results of this work as they explain some of its limitations and provide a perspective for future mapping studies of YT and similar platforms.

6.1 Challenges and threats to validity in mapping a video platform

We now discuss findings from the first research goal introduced in Section 1: *RG1. To develop automated support for mapping studies of YT.*

In this paper, we have shown a possible way of performing such a mapping study. Our approach focuses on automated means to manage the scale of the platform. There are many challenges to overcome and many other ways a mapping study could be performed. Even a completely manual mapping could be imagined, provided that the topic under study is sufficiently narrow. For topics such as the one in our work however, it almost certainly needs to be done in an automated way to deal with the scale and dynamics of YouTube, and we have seen that this can yield interesting results about the division of content over topics on the platform.

When designing and executing our mapping method (as described in Section 4), we encountered challenges in retrieving videos for analysis. Overall, making a study like this yield results at all requires to make a number of assumptions, thus yielding a number of threats to internal and construct validity. Throughout this study we have documented these choices and justified them as best we can and reflect in the following on our findings. We describe each of them below and discuss implications for this mapping study and future ones of the same sort.

6.1.1 Challenges in study design. A first category of challenges is related to the study design.

Finding the right granularity. It is not straightforward to identify the feasible level of granularity for a mapping study on YT. As with any mapping of the literature, a too narrow search string would result in too few videos and a too broad search string would result in too many videos. In contrast to the literature however, it is more difficult to get a reliable overview of the real total number of results of a particular search, and an intuition of an appropriate level of granularity is not present due to little experience and references in the literature. We have collected two datasets at different levels of granularity to explore a suitable level for future work. However, we cannot give a definitive recommendation.

Topic selection. Search strings for mapping studies differ based on their target platform, utilizing the specifics of each platform to maximize the possible results. Academic databases provide field-specific indexing, controlled vocabularies, and full Boolean logic for

including and excluding certain words from the search, allowing researchers to build a single, highly expressive query that maximizes precision across well-defined documents. In contrast, YouTube's public API applies each keyword query indiscriminately to titles, text descriptions, and user-supplied tags. It lacks proximity operators and nested Boolean clauses, and returns only a limited subset of videos the algorithm deems worthy to watch.

This means that broad terms that can be safely refined syntactically in scholarly databases quickly saturate YouTube's narrow results window with popular but off-topic videos, while the absence of controlled vocabulary and the diversity of user tags further degrade how precise the search is. All of these reasons mean that researchers conducting a mapping study on YouTube need to work around this lack of structure by making their own video filtering methods, as we have shown in this paper.

Future research implementing similar filters should focus on comparing the effects of different ordering of the results as returned by the API. By using other filters than relevance, the results avoid the opaque implementation of the search engine and instead rely on objective metadata, at the cost of potentially finding fewer relevant videos.

Choice of search strings. Search strings naturally affect the number of search results. However, we found titles and tags are often inconsistent, leading to additional difficulty in composing an effective search string. A good search string needs to find an appropriate balance between being too detailed and too general. Recent methodological guidance for researchers who want to use YouTube as a data source warns that "*keyword inflation, synonym gaps and polysemous terms can materially change recall*" and recommends multi-term, iterative search strategies to curb this loss [28]. Empirically, Jeong's large scale audit of 16 thousand videos shows that more than half of all words in tags, titles and descriptions overlap, meaning that a single unnoticed synonym can hide large sections of content from keyword queries [15]. In our pipeline, every SWEBOK topic is represented by a search string. As a result, the effective recall ceiling is already lower than the headline *pageInfo.totalResults* numbers reported by the API.

Restrictive filtering. As part of our selection criteria, we selected only English language videos published in 2015 or later. Internal logs (see supplementary material [29]) show that the language filter removes about 45% of the 500 candidates per page, and the publishing date threshold a further 30%. After all filters are applied, about 20% of the raw search results are left. This means that 80% of videos are discarded due to our filtering strategy. For our exploration of available educational content, this is acceptable since we still find relevant videos, but future work should be done to study what the effect of filtering out these videos is.

6.1.2 Challenges due to the available data on the platform. Secondly, there are challenges due to the availability of data on YT, and the way the YT data API works.

Availability of videos. Beyond the design choices for searching, the results are also impacted by limited availability of videos. Not all videos that appear in the search results are available for retrieval, they can be deleted, set to private, be region-blocked for the API's default region (US), or possibly be age-restricted. Such items are omitted from the paged response. These mismatches explain part of the discrepancy between the API's headline counts of *totalResults* and the stable IDs we ultimately store. This is one factor that complicates reliance on *totalResults* as a reliable indicator of the number of real available search results from a query, thus complicating the above challenges, too.

Inconsistent video metadata. When videos are available, their metadata may still present challenges for analysis. In contrast to a traditional population of publications in typical systematic mapping studies where publications follow a common format, YouTube is unstructured. Video titles can be specific and to-the-point (e.g., "Binary Search Explained") to rather broad and non-descriptive (e.g., "TOP 5 Coding Hacks THEY Don't Want You To Know"). Video metadata often misses tags or they are misspelled. Moreover, the same concept may be filed under different categories such as *Education, Science & Technology*, or simply *People & Blogs*. This redundancy inflates the *totalResults* value that the API returns. The number of results should be seen as an upper bound rather than an exact number. A mapping study as ours needs to manage these shaky grounds and cannot rely on the solid foundations of bibliographic databases.

Recommender algorithm impact. YouTube's video recommendation algorithm orders search results in a non-transparent way. In a conventional systematic mapping study, the researcher records the total number of documents returned by each search string. Even when a video contains the exact parameters we look for, the search endpoint does not return a full dump of results. Instead, it yields the n items that the ranking layer scores highest by weighting freshness, engagement signals, and click-through history [8]. This means we cannot know how many potentially relevant videos lie beyond the returned window, nor can we guarantee that repeated calls with pagination tokens will eventually exhaust the result set, the algorithm may keep reordering or filtering items as recommendations change. As a consequence, no precise number can be given for how many videos exist on a given topic and this is one of the reasons we limit apriori our dataset to 500 videos on each topic.

Moreover, there is a risk of missing relevant videos. Because each of our SWEBOK-inspired search strings is broad, the ranking filter skews the results, popular generalist videos are listed above niche but topic-wise relevant ones. To improve the recall, we focused on *relative coverage* rather than on absolute number of videos. We compare the distribution of retrieved videos across SWEBOK topics and interpret gaps as *lower bounds* or under-representation.

A possible solution to this narrow scope could be to use multiple similar search strings for collecting videos on a topic instead of just one, like our study did. Using many search strings to collect videos on the same topic could collect a more representative sample pool and mitigate recommender bias, at the cost of expanding the search space.

YouTube data API limitations. API pagination limits and video response caps impose practical limits on this kind of study. The API paginates at 50 results per call. While the search result value *pageInfo.totalResults* can display six-digit counts, developers' documentation and our own tests confirm that only the first 500 videos are actually accessible by page token for a given (query, channelId)

tuple. This means that a query can result in at most 500 items for further processing, regardless of any other number of found results. The API's daily quota of ten thousand tokens imposes a practical limit to what data can be collected and how much of it can be collected. In practice, we circumvented this by collecting the data for this paper over multiple days. Collecting at most 500 videos per topic was a pragmatic decision and a threat to the validity of our study. We consider it as acceptable because our mapping is rather exploratory in the first place.

Finally, the API returned only coarse approximations of the number of videos available for a given search query, which limited the precision of our estimates regarding the total volume of content on specific topics. Despite these constraints, we used the available metadata to map software engineering content on YouTube, albeit with trade-offs in granularity and scale.

Snowballing. Mapping studies of the literature make use of forward and backward snowballing techniques [20] to gather further relevant papers that did not yet appear in the initial search. It is an interesting challenge to try to implement a similar mechanism on a mapping study of YT. Of course, citations are not part of this dataset, but substitutes could be imagined in the future. One option could be to look at playlists, considering that if a video appears in a playlist, the other videos on that playlist may be of interest too. Another option could be to consider explicit references to other videos in the description of a found video. These ideas are not yet implemented in the current pipeline but could be an interesting direction for future work.

6.2 Trends in software engineering topics on YT

We now discuss findings from the second research goal introduced in Section 1: *RG2. To discover trends in videos about software engineering topics on YT.*

We have seen video upload trends with more videos in the dataset from recent years (in Figures 2 and 4). This data suggests a focus on early stages of the development life-cycle, with videos on Software Requirements being the most frequently occurring in the dataset. The data also contains terms which describe individual tools commonly used in practice, implying that there is also a strong degree of tool-assisted concepts being discussed. Videos about specific tools are more concrete and therefore easier to discuss and possibly better appreciated by persons starting out and learning by example, rather than learning at the conceptual level.

To further understand the trends in these videos, we analyzed what type of YT channels publish them, and what type of content these channels focus on. We calculated a specialization score based on video relevance to determine the top five channels in each category. The specialization score is built up from four weighted elements: the average relevance score of videos on the topic (40%), the maximum relevance score of videos by the channel on the topic (30%), the ratio of highly relevant videos (relevance score greater than 5.0) as portion of the total number of videos by the channel on the topic (20%), and a volume score for the number of videos by the channel on the topic (10%).

The results of this scoring mechanism are shown in Figure 6; from it, we can identify most high-scoring channels as dedicated education channels. We have not specifically sought out such content, but it has emerged automatically from the analysis. Initially, we have considered YT as the search platform and not considered more formal education content such as MOOCs. Nevertheless, the videos in the collected dataset may be publicly available educational material of this kind.

Our specialization score is one way to visualize these results and there are certainly other ways to look at them. Our main goal with this visualization is to show that our topic analysis and automated mapping can result in interesting insights about the coverage of the searched topics on YT.

6.3 Reflections

Given these results of the mapping study, we can now reflect back to the starting point of the paper. What did we learn about what content is available for citizen developers on YT? We see that there is a broad set of videos that provides a solid basis for learning software engineering concepts. These videos are of sufficient relevance to occur in our dataset and seem of sufficient quality to serve as educational material. Whether these videos are indeed used as such is a different question, that may be possible to answer in future research.

Other open questions from this reflection relate to what could be concluded with more detailed analysis of the videos themselves. Our current analysis is focused on textual transcripts, and therefore we miss rich information contained in visual aspects of the video. Future research could focus on a very narrow set of these topics and, instead of mapping them as we did, focus more on understanding what a good video is for this target audience. This would require manual analysis.

Related to the popular topics of the videos, we observe that the current videos focus on requirements and maintenance; why is that? Are these perhaps common starting points for extending learning beyond just programming? Or are these perhaps well tool-supported aspects that therefore attract more views? There is more research that can be done about the observed video publication trends, too. The current numbers in the dataset could be influenced by the search algorithm having a preference for more recent videos. This may skew the results. Another way of looking at this is that it could be interesting to perform several mappings, each with different time windows. This could be used to reconstruct a timeline of when a certain topic emerged as popular in videos.

Lastly, we consider the potential impact of studies such as these. Beyond just our interest, we believe that one of the responsibilities of our community is to improve the general knowledge of software engineering and access to educational materials beyond dedicated software engineering programs. Informal sources such as YT videos can greatly contribute to this mission. Indeed, the citizen developers who answered our questionnaire indicated to use YT for learning, and indicated how they search for content. It is thus relevant to extend this work to understand what makes videos useful for citizen developers, and where are gaps in topics that are not yet sufficiently covered. From the educational perspective, what could be considered in addition is how to reach citizen developers with "unknown unknowns", that is, aspects that we consider important

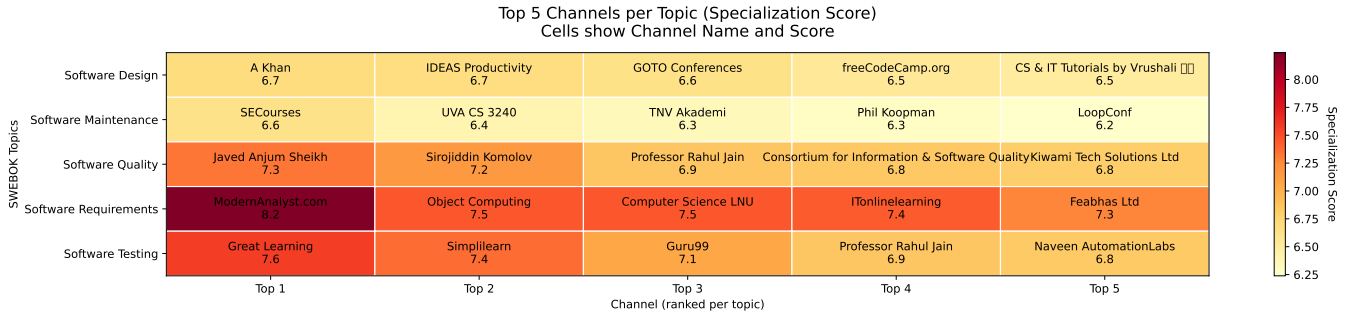

Figure 6: Top five channels per topic, and their specialization scores, determined by video relevance.

about software engineering but that the these citizen developers may be completely unaware of.

7 CONCLUSION

In this paper, we reported on a mapping study of YT with the goal of finding educational software engineering content aimed at citizen developers, to understand how they learn about crucial software engineering topics, beyond programming alone. The topics in our mapping were selected from the Software Engineering Body of Knowledge (SWEBOK), based on the input of 52 citizen developers that replied to our questionnaire survey. This resulted in topics for two mapping studies, one for high-level and one for low-level analysis, by which we refer to the level of granularity of the search terms.

The core objective of this work was to identify the informal software engineering educational content available for citizen developers on YT. Due to the scale of the platform, this was not possible to achieve manually and therefore we aimed to provide automated means for this and future mapping studies of YT. A custom script was developed for collecting relevant videos and generating transcripts in order to assess the relevance of each video to the chosen topic. Language processing techniques were used in the transcription process. All implemented scripts are documented and available in the supplementary material to this paper [29].

A significant part of the automated pipeline was to automatically assess the relevance of a given video for the searched topic. We developed our own analysis mechanism and performed a manual validation of this video relevance scoring.

Our mappings showed that there is a broad selection of educational software engineering content on YT. Across five high-level SWEBOK areas the number of videos in our dataset uploaded after 2020 is clearly increased with respect to older videos. Most growth is concentrated in the topic of *Software Maintenance*, while *Software Testing* and *Software Quality* receive far less attention. When we further explore more detailed topics in the low-level map, we see the same imbalance inside a single knowledge area. Videos about managing and analyzing requirements occur most often, whereas specification and functional-detail tutorials are more scarce. The automated transcript analysis that we implemented in our pipeline indicated that videos focus on a small cluster of practical tasks and rarely branch into deeper or longer-term quality topics.

While our results give a first indication of the landscape of educational content available to citizen developers, we also discuss significant limitations to this and other automated mappings. The scale of the platform, as well as our high-level dataset, show that there is a strong base of software engineering videos. Our results also show that high-level queries return more videos than even an automated mapping study can analyze meaningfully. Conversely, our mapping at lower level of granularity contained more useful insights, due to a more manageable volume of results in therefore more clearly discoverable trends in topic coverage and video relevance. Part of our results are suggestions for future mapping studies, and we hope that our experiences can contribute as input to them, too.

In conclusion, this mapping of YT is a first step towards getting more insight into the available educational software engineering content, and provides automated means for future mapping studies on this and other topics. We consider our results as mixed; while we show the concrete results of mappings at two levels of granularity, we also describe in detail the challenges we encountered in doing so, and the limitations for such studies. We hope that these, and the corresponding suggestions for future studies are still interesting for the community.

This paper can serve as a foundation for future mapping studies of YT for other topics. It is possible to run such studies for longer time or to apply for more generous API limits, but this will not fundamentally change the type of results that are obtainable, just the volume that is possible to analyze. A larger volume could allow for a clearer view of the overall landscape of content available on the YT platform, with the ultimate goal of identifying more distinct trends. The methodology remains tricky though, this paper shows several challenges we encountered and that would need improvement in future studies.

For future work on the topic of citizen developers' learning, we believe it would be highly valuable to investigate the other side as well, e.g. user studies on how citizen developers engage with this content to validate its educational relevance.

REFERENCES

[1] Björn Binzer and Till J Winkler. 2022. Democratizing software development: a systematic multivocal literature review and research agenda on citizen development. In *International Conference on Software Business*. Springer International Publishing, Cham, 244–259. https://doi.org/10.1007/978-3-031-20706-8_17

[2] Steven Bird, Edward Klein, and Ewan Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Sebastopol, CA.

[3] David M. Blei. 2012. Probabilistic Topic Models. *Commun. ACM* 55, 4 (2012), 77–84.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.

[5] YouTube Official Blog. 2025. *YouTube for Press*. YouTube.

[6] Souti Chattopadhyay, Thomas Zimmermann, and Denae Ford. 2021. Reel life vs. real life: how software developers share their daily life through vlogs. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Athens, Greece) *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 404–415. https://doi.org/10.1145/3468264.3468599

[7] Kenneth W. Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics* 16, 1 (1990), 22–29.

[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 191–198.

[9] Marc Faddoul, Guillaume Chaslot, and Hany Farid. 2020. A Longitudinal Analysis of YouTube's Promotion of Conspiracy Videos. https://arxiv.org/abs/2003.03318. arXiv:2003.03318.

[10] Lee Fallin. 2021. Teaching academic software via YouTube videos in the Covid-19 pandemic: potential applications for learning development. *Journal of Learning Development in Higher Education* -, 22 (Oct. 2021), 1–5. https://doi.org/10.47408/jldhe.vi22.679

[11] Vahid Garousi, Görkem Giray, Eray Tüzün, Cagatay Catal, and Michael Felderer. 2019. Aligning software engineering education with industrial needs: A meta-analysis. *Journal of Systems and Software* 156 (2019), 65–83. https://doi.org/10.1016/j.jss.2019.06.044

[12] Fabio Giglietto, Luca Rossi, and Davide Bennato and. 2012. The Open Laboratory: Limits and Possibilities of Using Facebook, Twitter, and YouTube as a Research Data Source. *Journal of Technology in Human Services* 30, 3-4 (2012), 145–159. https://doi.org/10.1080/15228835.2012.743797

[13] Google Developers. 2024. YouTube Data API v3. https://developers.google.com/youtube/v3 Accessed: 2025-04-05.

[14] Eduard C Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitza Guzman, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, et al. 2017. The crowd in requirements engineering: The landscape and challenges. *IEEE software* 34, 2 (2017), 44–52. https://doi.org/10.1109/MS.2017.33

[15] Wooseob Jeong. 2009. Is Tagging Effective? – Overlapping Ratios with Other Metadata Fields. In *Proceedings of the 2009 International Conference on Dublin Core and Metadata Applications*. 31–39. https://doi.org/10.23106/dcmi.952109539

[16] Barbara A. Kitchenham, David Budgen, and O. Pearl Brereton. 2010. The value of mapping studies: a participantobserver case study. In *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering* (UK) *(EASE'10)*. BCS Learning & Development Ltd., Swindon, GBR, 25–33.

[17] Mark Ledwich, Anna Zaitsev, and Anton Laukemper. 2022. Radical Bubbles on YouTube? Revisiting Algorithmic Extremism with Personalised Recommendations. *First Monday* 27, 12 (2022). https://doi.org/10.5210/fm.v27i12.12552

[18] Kostantinos Papadamou, Savvas Zannettou, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Michael Sirivianos. 2021. "It is just a flu": Assessing the Effect of Watch History on YouTube's Pseudoscientific Video Recommendations. arXiv:2010.11638 [cs.CY] https://arxiv.org/abs/2010.11638

[19] Minsu Park, Mor Naaman, and Jonah Berger. 2016. A data-driven study of view duration on youtube. In *Proceedings of the international AAAI conference on web and social media*, Vol. 10. 651–654.

[20] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

[21] Juan Ramos. 2003. Using TF-IDF to Determine Word Relevance in Document Queries. CS Department, Rutgers University, Technical Report.

[22] Bernhard Rieder, Òscar Coromina, and Ariadna Matamoros-Fernández. 2020. Mapping YouTube: A quantitative exploration of a platformed media system. *First Monday* 25, 8 (2020), Article–number.

[23] Gerard Salton and Christopher Buckley. 1988. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management* 24, 5 (1988), 513–523.

[24] Alexandra Schofield, Måns Magnusson, and David Mimno. 2017. Pulling Out the Stops: Rethinking Stopword Removal for Topic Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Mirella Lapata, Phil Blunsom, and Alexander Koller (Eds.). Association for Computational Linguistics, Valencia, Spain, 432–436. https://aclanthology.org/E17-2069/

[25] Claude E. Shannon. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27, 3 (1948), 379–423.

[26] Abdulhadi Shoufan and Fatma Mohamed. 2022. YouTube and education: A scoping review. *IEEE Access* 10 (2022), 125576–125599. https://doi.org/10.1109/ACCESS.2022.3225419

[27] Abdulhadi Shoufan and Fatma Mohamed. 2022. YouTube and education: A scoping review. *IEEE Access* 10 (2022), 125576–125599. https://doi.org/10.1109/ACCESS.2022.3225419

[28] Wuyou Sui, Anna Sui, and Ryan E. Rhodes. 2022. What to watch: Practical considerations and strategies for using YouTube for research. *Digital Health* 8 (2022). https://doi.org/10.1177/20552076221123707

[29] Nikola Vasović and Maša Pejović. 2025. *Supplementary material for the paper "Mapping educational software engineering content on YouTube"*. https://doi.org/10.5281/zenodo.17219939

[30] Hironori Washizaki (Ed.). 2024. *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0*. IEEE Computer Society, Los Alamitos, CA. https://www.swebok.org

[31] Ruoyu Zhang, Jennifer Shin, Kristine Schulz, Xiao Liu, Anjana Susarla, and Rema Padman. 2024. YouTube Video Analytics for Patient Education: An Exploratory Clustering of Obstructive Sleep Apnea Videos. In *MEDINFO 2023*. https://doi.org/10.3233/SHTI231167

[32] Ge Zhu, Juan-Pablo Caceres, and Justin Salamon. 2022. Filler Word Detection and Classification: A Dataset and Benchmark. arXiv:2203.15135 [cs.CL] https://arxiv.org/abs/2203.15135