On Mining Crowd-based Speech Documentation

Parisa Moslehi
Concordia University
Montreal, Canada
p_mosleh@encs.concordia.ca

Bram Adams
École Polytechnique de Montréal
Montreal, Canada
bram.adams@polymtl.ca

Juergen Rilling
Concordia University
Montreal, Canada
juergen.rilling@concordia.ca

ABSTRACT
Despite the globalization of software development, relevant documentation of a project, such as requirements and design documents, often still is missing, incomplete or outdated. However, parts of that documentation can be found outside the project, where it is fragmented across hundreds of textual web documents like blog posts, email messages and forum posts, as well as multimedia documents such as screencasts and podcasts. Since dissecting and filtering multimedia information based on its relevancy to a given project is an inherently difficult task, it is necessary to provide an automated approach for mining this crowd-based documentation. In this paper, we are interested in mining the speech part of YouTube screencasts, since this part typically contains the rationale and insights of a screencast. We introduce a methodology that transcribes and analyzes the transcribed text using various Information Extraction (IE) techniques, and present a case study to illustrate the applicability of our mining methodology. In this case study, we extract use case scenarios from WordPress tutorial videos and show how their content can supplement existing documentation. We then evaluate how well existing rankings of video content are able to pinpoint the most relevant videos for a given scenario.

CCS Concepts
• Information retrieval → Document representation • Software and its engineering → Software post-development issues.

Keywords
Crowd-based documentation; mining video content; speech analysis; Information Extraction; software documentation

1. INTRODUCTION
In traditional software development processes, software documentation has played a vital role in capturing information relevant to the various stakeholders and as assessment criteria for the maturity and quality of a software product and its underlying development processes. However, over the last decade, with the economical (e.g., globalization, open source projects), social (e.g., collaborative and agile work habits), and technological (e.g., Internet) changes in our society, new software processes (e.g., agile) have been introduced to deal with these ongoing changes.

What is common to these development processes is that communication and data become more important than formal and complete documentation. As a result of this paradigm shift, projects are now often left with no or very little documentation [1], or documentation that is often incomplete or lacking sufficient details (explanation and examples) to meet the needs of the various project stakeholders.

Users have started to resort to the Internet for help, with its wealth of information, to find documentation and explanations to support their current work context. On the other side, what users can find, are crowd-based documents that are written by many and read by many [2], with contributors being motivated by the fact that they can gain an online reputation and also learn more through the process of teaching to others [3]. However, this crowd-based information tends to be fragmented across hundreds of textual web documents like blog posts, email messages and forum posts, as well as multimedia documents such as screencasts and podcasts. While, for a given feature, the essential information likely will be common, optional information could be different across the documents, and documents could even contain contradictory or outdated information!

One way to help stakeholders find the information that they require for a given feature, is to automatically mine web documents and recover the relevant documentation. Recently, the MSR community has started mining unstructured repositories such as mailing lists, Q&A sites and Wiki pages by analyzing non-structured free form text. However, only limited work [3],[4],[5] exists on mining and analyzing speech and video files. In this paper, we are in particular interested in mining the speech component of YouTube videos, since those typically contain the rationale and major insights about what a video is trying to achieve, complementing the visual cues.

Hence, this paper proposes a methodology to automatically transcribe and analyze the content of the transcribed text using various Information Extraction (IE) techniques to both supplement existing software documentation and to help users focus on the relevant sources. As part of our evaluation, we present a case study in which we extract use case scenarios from the speech of WordPress tutorial videos. For the second part of our case study, we evaluate how well existing metrics of videos are able to rank videos according to relevancy for a given feature or use case.

This paper extends traditional MSR approaches to include the mining and analysis of speech components of video material related to software products. A further contribution of this work is to provide a methodology that allows the automated extraction of software documentation from speech content. The presented case study illustrates not only the feasibility but also benefits of mining the speech component of videos and how users in a software development or maintenance context can benefit from this "wisdom" of the crowd.

MSR'16, May 14-15, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2901739.2901771

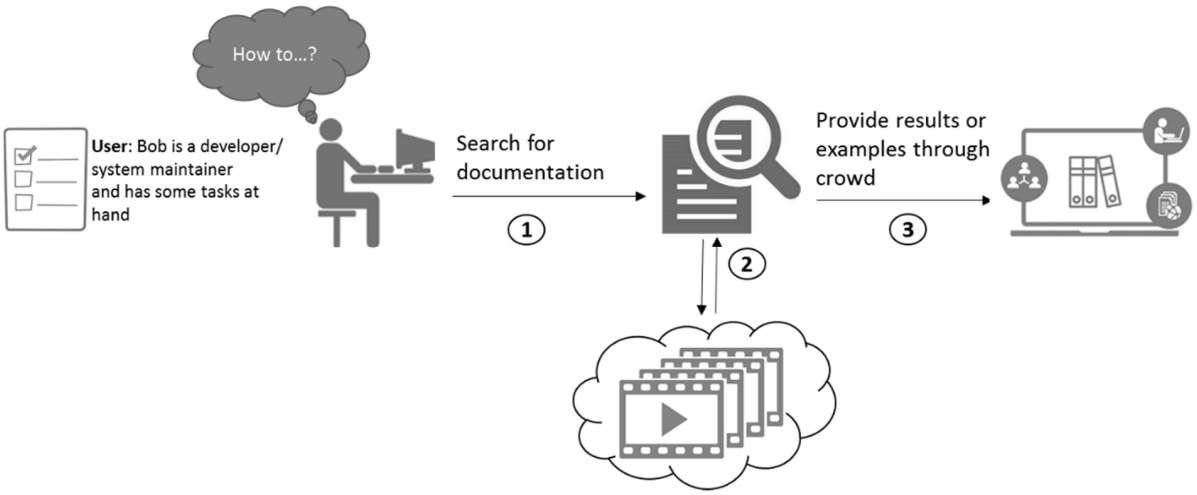

Figure 1. Motivating example

The paper is organized as follows: Section 2 lays out the motivation and research questions of our work. Section 3 describes our research methodology, data, and data extraction techniques, including a proof of concept implementation. Section 4 answers our research questions through an initial case study. Section 5 compares our work with some related works to ours. Section 6, discusses some threats to validity, followed by conclusions and future work in Section 7.

2. Motivation

In this section, we are presenting the motivation behind our research and the research questions, i.e., why we are interested in performing IE from screencasts (and their speech component).

Our research is motivated by the fact that developers and maintainers often need to resort to the available software documentation to be able to comprehend a problem and to identify a solution in order to complete their task at hand. However, especially in the context of open source projects or agile development processes, existing system documentation tends to be incomplete, inconsistent or does not provide clear enough examples on how to use a particular feature or solve the given problem [17]. Therefore, users typically resort to additional online resources for help [2], [3], usually created by users not directly involved in the development of the actual product, also often referred to as "the crowd". These online resources can be published and made available in many different digital formats such as: wikis (e.g., Wikipedia), Q&A repositories (e.g., StackOverflow) or tutorials published as videos or podcasts (e.g., YouTube or Vimeo).

Figure 1 shows an example where user Bob, an open source enthusiast, wants to modify a feature in the Akismet WordPress plugin for comment moderation. Bob, who is unfamiliar with the innards of the plugin, first resorts to the available software documentation to understand the context, how the feature he wants to change is designed as well as the purpose of the current implementation. However, Bob is unable to find any relevant documentation describing the feature of interest. He recalls that he has seen in the past, several how-to videos and blogs on the Internet, describing the various features of the plugin and decides to search for relevant videos describing the feature he is currently working on. One particular kind of video that provides concrete information and examples to users and developers such as how to use or troubleshoot a particular application, are so-called screencasts [3]. These are videos that show the user interface of a software system as someone is using it to complete a particular use case scenario, step by step. Most of the screencasts have accompanying narrative, where a person explains what (s)he is doing, and/or on-screen annotations and other cues to highlight or summarize important information.

However, several challenges exist when using such screencasts published by the crowd as a source of documentation [3]. We categorized these challenges as follows: a) **Relevance:** Although screencasts and crowd-based documentation contain useful instructions on how to accomplish a specific task, developers may spend significant time trying to identify what video to watch. The relevance from a developer's perspective depends on how closely a video addresses the task at hand. Developers often have to rely on manual browsing through videos to see whether the video actually contains content that is relevant to their needs or to skip non-relevant parts of videos. b) **Quality:** Even if a video is relevant, its quality might vary and is unknown to the users prior to watching it. For example, the video might not cover the complete use case at hand, or video and/or speech could be blurry; c) **Quantity**: Depending on the popularity of the topic, not only a large number of videos might exist, but also their length and content might vary significantly. Existing search engines rely on the indexing of user provided titles and descriptions, or words recovered from automatic speech-to-text, but do not consider the actual content (semantics) in their search and ranking.

In this research, we aim to address some of these challenges in guiding developers to mine and identify screencast content that might be relevant to their given work context. More specifically, we focus on the automated mining of the speech part of screencasts, by transcribing the speech content and applying IE to mine the content of these transcriptions. We focus on the speech part, since (1) the speech often contains the rationale and guidance of what is happening on the screen, and (2) analysis of the video part requires advanced image processing in order to provide meaningful information [4], [5].

More specifically, we will be answering the following two research questions:

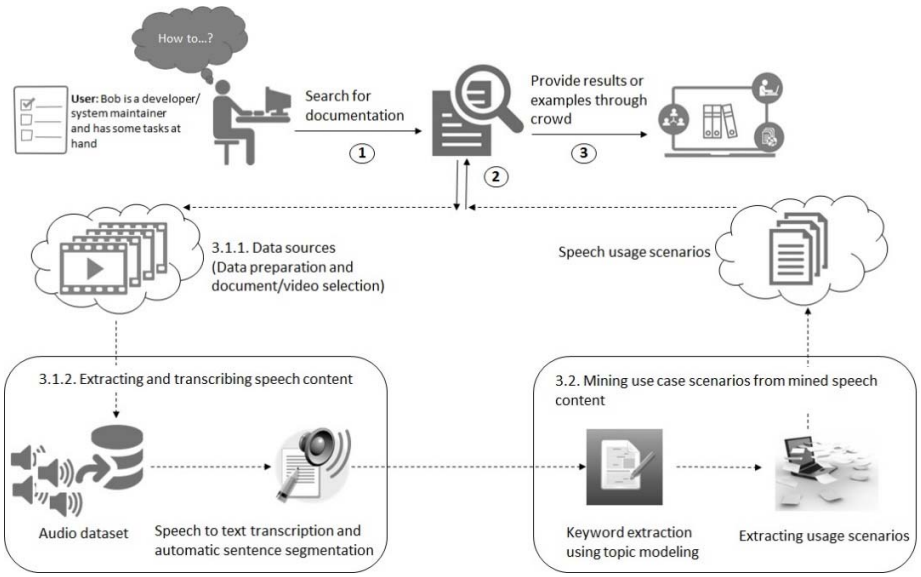

Figure 2. The proposed methodology to mine the speech part of screencasts.

RQ1. Is it possible to derive an automated approach to extract software engineering related information from the speech part of crowd-based videos to supplement existing documentation?

RQ2. How well do existing screencast ranking mechanisms perform in terms of relevancy of the mined speech content?

In fact, in RQ1 we will evaluate whether our approach can be used to extract and recover use case scenarios from the speech part of how-to screencasts published on YouTube[1]. For RQ2, we rank screencasts based on how complete their coverage of a given use case scenario is (as mined in RQ1), then evaluate which video-related metrics provide the closest ranking.

3. Methodology

This section presents an overview of the proposed methodology for mining the speech part of screencasts to recover use cases. Given a screencast on, for example, how to create a blog post, we want to obtain a sequence of essential and optional use case steps (e.g., 1. Click on the "new button", 2. Type the title of the blog post, 3. (Optionally) change the font, 4. Type the content of the blog post). By mining a corpus of screencasts related to this scenario, steps that occur across all screencasts can be interpreted as essential, while those appearing only in some screencasts as optional.

We split the methodology in two parts, i.e., (4.1) a general part applicable to any kind of mining speech data, and (4.2) a part specific to recovery of use cases from screencast speech. We also provide a proof of concept implementation of our methodology.

3.1 Mining Speech Data from Screencasts

In what follows, we describe the automated approach that we derived for mining the speech part of crowd-based screencasts. As mentioned, this part of the methodology is independent from how the mined data will be used.

3.1.1 Data Sources

The data sources used by our approach can be any podcasts or screencasts available online that target a specific software project or even feature. Such data typically can be found on YouTube, Vimeo or on a project's wiki or web site, and usually contains a how-to or walkthrough tutorial, or sometimes workarounds for known bugs. It should be pointed out that, any video source would be applicable as long as it has a speech component without disturbing music or background noise.

3.1.2 Extracting and Transcribing Speech Content

Before any analysis can be done, we first need to extract the spoken text of the video as a speech file using standard speech extraction tools. We ignore the image part of the video in the remainder of this paper, whereas other researchers recently have started mining this information [4], [5].

Given the speech part of a video, we then need to transcribe the downloaded speech files using a Speech-To-Text (STT) tool to obtain an automatic transcription of the spoken text. The availability of an STT transcription tool that can produce outputs that are accurate enough and have a low enough error rate is an essential requisite. While transcribing speech files in general is a challenging aspect, most STT tools are capable of supporting some standard English dialects (mostly native US or UK English). In order to improve the transcriptions of spoken text, the majority of existing STT tools (e.g., Dragon NaturallySpeaking[2]) rely on an initial training step, during which the tool can be trained to recognize a particular voice (dialect) of a user.

However, transcribing speech content from crowd-based video portals introduces additional challenges. Among these challenges are that videos (and therefore also their speech components) are created and published by many different users with cultural, intellectual, communication and linguistic differences. While processing crowd-based speech sources (where by definition the presenter is not one single person), STT tools will have to deal with the resulting variety of English dialects (e.g., native speakers, non-native speakers with accents or grammatical errors). Unfortunately, a training phase (as Dragon is doing) is not feasible, since the videos are crowd-based. In order to deal with this uncertainty, some tools provide as part of their transcription also indicators about the accuracy and quality of their transcription. Given that the quality of

[1] https://www.youtube.com/

[2] http://www.nuance.com/

the STT output effects our methodology, we eliminate transcriptions that are flagged by an STT tool to be of potentially low quality using a confidence threshold of 0.7 (section 3.3.1).

3.1.3 Segmenting Transcribed Text

Another major challenge in STTs besides grammar and dialect, is the need to extract punctuated sentences instead of a bag of words. Punctuation, often also referred to as sentence segmentation, is an essential aspect to allow for further processing of the transcribed text through IE systems (e.g., GATE[3], OpenNLP[4]). IE systems analyze text in order to extract information about pre-specified types of events, entities or relationships, where the order in which the information is presented matters. In order to be able to extract this information, IE tools require punctuation and newline characters for sentence splitting and text segmentation.

Nevertheless, STT tools provide only very limited support for punctuation in their transcriptions, mostly relying on fixed time thresholds for speech pauses. However, without taking into consideration a user's specific speech and speed patterns, sentence segmentation of spoken text becomes inherently difficult due to the incorrect punctuation in the transcribed text resulting often in no logical sentences. Furthermore, users tend to use informal grammar and sentence structures when narrating their videos with spoken text. This and any grammar and spelling issues generated by incorrect transcription further reduces the chances to correctly identify sentence boundaries. Figure 3 illustrates these challenges.

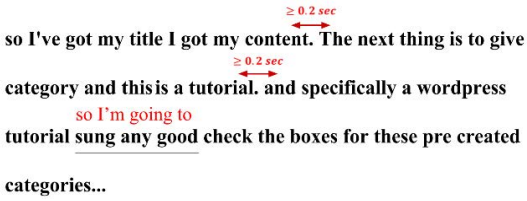

Figure 3. (a) Threshold based sentence segmentation based on speech pauses. (b) Imprecisions caused by grammar and transcription errors.

In our methodology, we take advantage of timestamps commonly provided by modern STTs for each word in the transcribed text, then calculate the pauses, considering the baseline approach in [6] and their *mean* and standard deviation (*std)* values for each file to adjust the punctuations based on the speaking pace using these non-acoustic features. As part of an initial experiment, we observed that using an adjustment factor $\alpha=3$ for the *std* produces the best sentence segmentation results:

$$(1)\ mean + \alpha \times std$$

3.2 Mining Use Case Scenarios from Mined Speech Content

This section explains our methodology to mine use case scenarios from the sentences mined from the screencast speech.

Essential use case steps by definition should be common to all transcribed videos, while optional ones should be specific to a subset of them. This means that in principle we could apply any IE techniques such as, text classification or clustering on the corpus of transcribed speech [9], to extract actual use cases from the screencasts.

However, common IE approaches for textual data such as dependency parsing [7], [8] or language models [9] are not suitable for our dataset, since the grammar and syntax of sentences in spoken text differ significantly from those of traditional written text and there is not enough domain-specific training data for developing a language model. This situation is further complicated by additional errors introduced by STT tools, during the transcription of the spoken text.

Based on our review of existing IE approaches, we selected topic modeling [10] to identify the topics that are shared amongst all videos for a given system feature. Topic models provide statistical information related to sets of words ("topics") that occur together often enough to represent a semantic relation [11]. For example, in a newspaper the "sports" topic consists of sports-related words that co-occur across most of the sports-related articles. Given that all essential steps of a given use case should be mentioned by each screencast related to that use case, we conjecture that there should be at least one dedicated topic (group of words) containing all essential steps.

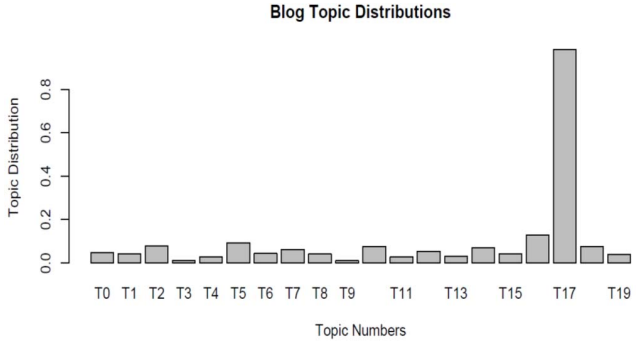

Figure 4. Distributions of topics for each topic model for the create *Blog* topic.

For example, the barplot in Figure 4 shows an example of the distribution of the group of words describing a topic across the transcripts of the use case "Creating a Blog". A higher value for "Distribution" means that the topic occurs in more transcripts and hence is shared by more videos. We would automatically select for each use case the topic with the highest distribution value (in this example it would be topic *T17*). That topic is assumed to contain all essential steps and some of the optional steps.

now when I point to the post **button** it gives me little fly out menu.
and I'm gonna **click** on **add** new it's something you'll get familiar with because this will be one of the most common activities for you inside of **WordPress** so here's the post.

Figure 5. Sentences with topic keywords (highlighted), corresponding to essential steps of a use case.

Finally, in order to reconstruct the individual use case steps from the selected topic, we first identify the 20 words of the topic that are the most common in the analyzed screencasts. These represent the topic's "keywords". We then use regular expressions to identify in each screencast's transcription all sentences containing these keywords (Figure 5). For a given screencast, the corresponding

[3] https://gate.ac.uk/

[4] https://opennlp.apache.org/

sentences are the essential and optional use case steps discussed by the screencast. Other sentences can be filtered out.

Since each screencast results in a sequence of extracted use case steps, all that is left is to rank the resulting screencasts based on the relevance of these steps. Coming up with a novel ranking mechanism is outside the scope of this work, instead in RQ2 we will evaluate existing ranking mechanisms in terms of relevancy.

3.3 Proof of Concept

To illustrate the applicability of our approach, we introduce a proof of concept implementation of our methodology, which we then use in the next section, for case studies to address our two research questions, introduced earlier in the paper. The following sections describe in more details technical and implementation considerations for creating our proof of concept implementation.

3.3.1 Data preparation and transcribing screencasts

For the automated transcription of speech content of videos, we had to select an STT tool that was suitable to deal with the challenges associated with analyzing speech content from video material created and published by many different users. For the tool selection process, we used several evaluation criteria, including: precision, recall, support for different English dialects (e.g., US, UK and Indian English dialects), scalability, provision of metadata (e.g., timestamps, accuracy score for transcript) and ease of use. As evaluation data we used one screencast, which we manually transcribed to establish a baseline.

Table 1. Comparison of the output of STT tools using British dialect, length of the transcript (in terms of characters), precision, recall and f-measure

Tool	Length	Precision	Recall	F-Measure
Dictation[5]	5376	0.75	0.66	0.70
Google API[6]	6458	0.75	0.75	0.75
IBM Watson[7]	7745	0.75	0.88	0.81
Speechlogger[8]	6971	0.74	0.73	0.73
Speechpad [9]	6415	0.70	0.74	0.72

Table 1 shows the five analyzed STT tools, consisting of commercial products by top vendors in the field, as well as how they performed on our evaluation. First, we manually transcribed the screencast, then determined for each tool how many words they yielded in their transcription, how many of those words were correct (precision), and how many of the manually identified words they were able to identify (recall). The F-measure then provides a combined measure of precision and recall, the higher the value the better both precision and recall are. We found that IBM Watson obtained the best recall (more words transcribed correctly), while precision was similar to the other tools. Hence, we decided to use IBM Watson.

Note that our evaluation did not consider the order of words into sentences. However, as shown in Figure 6, the sample JSON output of the IBM Watson STT provides metadata that is useful for the later processing steps, like sentence splitting and filtering of the transcripts. This was another major reason to opt for this STT tool.

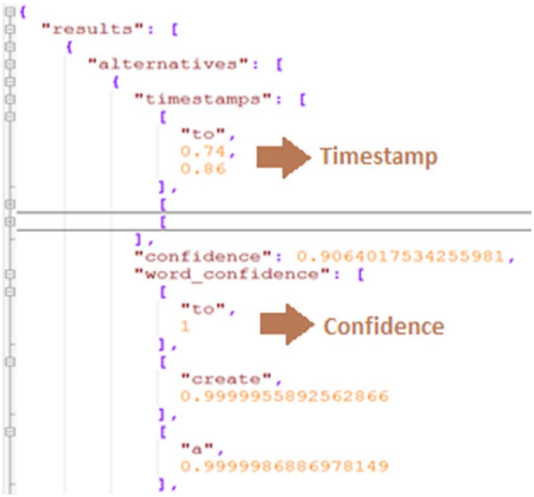

Figure 6. Sample JSON output created by the IBM Watson STT tool.

In the JSON file, *Word_confidence* corresponds to the STT's confidence level, according to which the tool estimates whether the transcribed words were correctly recognized. Based on manual analysis of the transcripts generated by the STT tool, we observed that a transcript with an overall score below a threshold of 0.7 contains too many errors to be useful for further processing. For our proof of concept implementation, we therefore eliminated all transcripts with a confidence score below the 0.7 confidence threshold.

While the IBM STT tool provides some very basic sentence splitting, this resulted in very few, extremely long sentences. Therefore, we performed the additional sentence splitting processing of formula (1). For this splitting process, we took advantage of the "timestamps" meta-data provided in the output of the IBM Watson STT. The tool provides the start and end time of each word in the speech file's transcription, which allows us to calculate the pauses between the words in the spoken text. We use these pauses in connection with the sentence splitting formula (1) to calculate the *mean* and *std* values of the pauses, for each file. Using this approach for sentence splitting, the number of sentences increases on average by a factor 5.

3.3.2 Information extraction

To identify topics across the transcripts, we used the standard MALLET tool[10] for topic model analysis. We evaluated different configuration settings, with the number of topics set to 5, 10 and, 20 topics for each use case and manually compared the granularity of the results. The analysis showed that, for our purposes, the configuration with 20 topics produced the best results. As a result of the extraction process, we then chose the topic with the highest distribution value across the corpus of screencasts of the use case under study.

4. Case Study

In what follows, we present results from a case study that we conducted to evaluate the applicability of our methodology and to answer our two research questions introduced earlier.

[5] https://dictation.io/
[6] https://www.google.com/intl/en/chrome/demos/speech.html
[7] https://speech-to-text-demo.mybluemix.net/
[8] https://speechlogger.appspot.com/en/
[9] https://www.speechpad.com/
[10] http://mallet.cs.umass.edu/

First (RQ1), we evaluate how a fully automated approach using IBM Watson STT performs. Then (RQ2), we evaluate whether existing screencast-related metrics would be good indicators of the relevancy of a screencast towards a specific use case.

Dataset: For our case studies, we created a dataset based on how-to screencast videos for the WordPress CMS, published on YouTube. We selected WordPress as our study subject, since it is one of the most widely used, mature and documented open source CMS tools. Furthermore, given the popularity of WordPress and its open source nature, a large number of how-to screencast videos exist on YouTube, covering most of the WordPress use cases and features. As our ground truth we use 5 topics selected from the WordPress online documentations[11] and made them publicly available[12].

Screencast selection was based on five keyword search queries (obtained from the official WordPress documentation) that we performed on YouTube: 1) "How to create a blog post in WordPress" (blog), 2) "How to change the font in WordPress" (font), 3) "How to password protect WordPress" (password), 4) "How to create a Gravatar in WordPress" (gravatar) and, 5) "How to add a feed to WordPress" (feed). We selected videos from top results and the recommendations made by YouTube. The length of the obtained screencasts varied from 1 to 10 minutes. Table 2 provides a summary of the datasets that we created for each of the queries. Of the initially downloaded screencasts, between 60% (blog and feed) and 83% (password) of the screencasts yielded a useful transcript with STT confidence of 0.7 or higher.

Table 2. The number of studied screencasts and transcripts for the five analyzed use cases.

Category	# screencasts	# transcripts	# transcripts filtered by confidence	% transcripts filtered by confidence
Blog	42	35	25	59.5
Font	27	21	21	77.8
Password	23	23	19	82.6
Gravatar	30	29	21	70
Feed	30	29	18	0.6

4.1 RQ1. Is it possible to derive an automated approach to extract software engineering related information from the speech part of crowd-based videos to supplement existing documentation?

Approach: After transcribing the downloaded screencasts (see Table 2), we experimented with different configurations of MALLET using 5, 10 and 20 topics. Table 3 shows, for each configuration, the maximum value for topic distribution. The lower this value, the more fine-grained the topics tend to be (covering less files). Hence, we are interested in higher values, however too high values might indicate too coarse-grained topics that contain everything-and-the-kitchen-sink. After manually checking the maximum distribution topics of the different configurations, we found that 20 topics, despite having the maximum distribution values, still were not too coarse-grained for our purposes. Hence, we used 20 topics in our case study.

Table 3. Topic distributions for each feature/topic model

Category	Maximum topic distribution		
	5 topics	10 topics	20 topics
How to create a blog post in WordPress	0.86	0.96	**0.98**
How to change font in WordPress	0.37	0.76	**0.98**
How to password protect WordPress	0.38	0.53	**0.63**
How to create a Gravatar in WordPress	0.39	0.26	**0.94**
How to add a feed to WordPress	0.33	0.66	**0.72**

Using the 20 topics setting, we selected, for each of the five use cases, the topic with the highest distribution value across all analyzed screencasts. If there are other highly distributed topics we would pick the highest one. Figure 7 shows these distribution values of the 20 topics for each use case. For example, for the *blog* use case, we would pick topic 17 as the one containing the most common topic keywords across all video transcripts for this use case. Using regular expressions, we then parsed the generated transcripts and extracted those sentences that contain at least one relevant topic keyword. Each of these extracted sentences corresponds to one use case step for this feature.

The processing time for the 5 datasets varied between 487.5 seconds for the *Password* dataset and 809.5 seconds for the *Blog* dataset. A detailed brake down is shown in Table 4.

Table 4. Detailed breakdown of processing times (seconds)

Dataset	STT	Topic Modeling	Regular Expression + JSON parser
Blog	790	1.5	18
Font	773	1.5	18
Password	458	1.5	28
Gravatar	582	1.5	30
Feed	683	1.5	19

Finally, based on the existing WordPress documentation, we manually derived a ground truth for each use case, i.e., the sequence of essential and optional steps that make up the use case. We validated these use cases by running them on an example installation of WordPress. Given these ground truth steps, we then manually calculated the precision, recall and F-measure of our approach. Precision in this context corresponds to the percentage of selected sentences in the transcripts that also occur in the ground truth, while recall refers to the percentage of steps in our ground truth that were found by the approach. The F-measure then is the harmonic mean of precision and recall, the higher it is, the higher the combination of precision and recall are.

[11] https://codex.wordpress.org/

[12] http://aseg.encs.concordia.ca/msr16/wordpress.zip

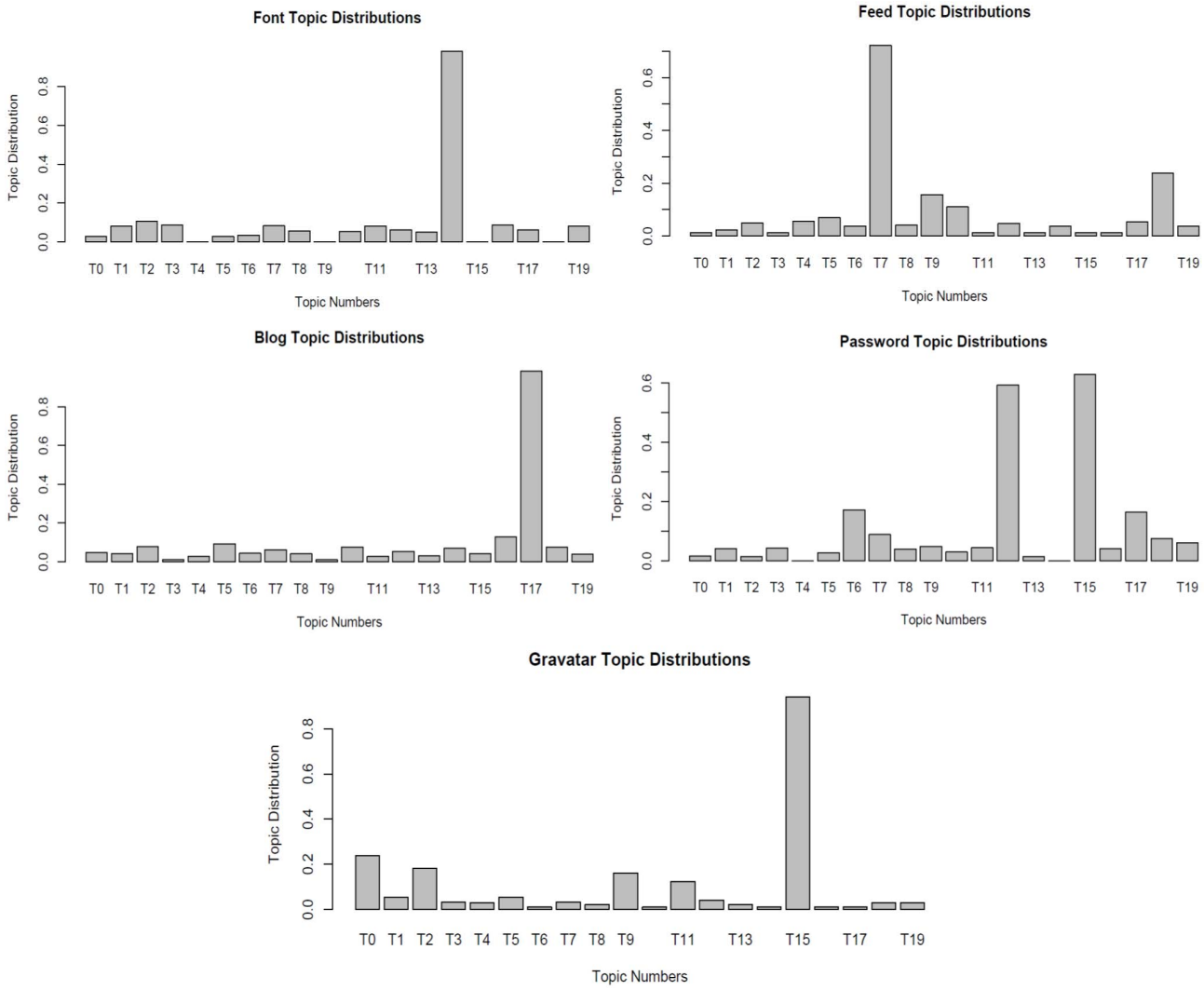

Figure 7. Bar charts showing the maximum distribution of topics for each topic model and use case data set.

Since evaluation of precision and recall required manual comparison of the ground truth and the mined use case steps (which is time consuming), we randomly selected 5 screencasts for each use case, making sure to include screencasts with many, few and a medium number of views. While we only have precision and recall for the resulting 25 screencasts, the topic models have been built based on all screencasts. Furthermore, note that we did not make a distinction between essential and optional steps for our calculations; hence missing optional steps will reduce the recall of our approach. This strict evaluation means that the obtained recall values are only a lower bound (ignoring optional steps would increase recall, and slightly reduce precision). Furthermore, we did not evaluate whether the mined use case steps occurred in the right order. From our informal evaluation, we observed that the order typically was correct.

Table 5 shows the corresponding precision, recall and F-measure values for the 25 manually evaluated screencasts.

Findings: Precision and recall both are high (median of 83.33% and 100%, respectively). Table 5 shows how the minimum precision value is 62.5% (Font), with most of the values 100%, implying that most of the use case steps suggested by the approach are correct (low number of false alarms). Recall, on the other hand, sometimes drops to 50%, meaning that only half of the relevant information (i.e., optional and essential steps) is recovered from a screencast. Overall, however, the 25 screencasts have a median "F-measure" value of 84.32%, indicating a very high performance.

Closer analysis of the use case steps produced by the approach showed that almost each screencast started with an introduction giving an overview of the content of the screencast, and ended with a recap. Both intro and recap were selected by the approach as being part of the use case steps, while our ground truth did not include it. Filtering out those sentences (typically at start/end of recovered use case steps) increases the median precision to 100%.

For each use case, at least one of the screencasts achieves a recall of 100%. This means that, theoretically, if one could somehow rank screencasts according to their predicted recall, users could just select the obtained use case steps of the top ranked screencast. If we assume such a ranking to exist, we could conclude that crowd-based speech documents would be a reliable source to enrich software documentation. RQ2 explores this in more detail.

Table 5. Precision/Recall of the 25 manually evaluated transcripts.

Feature	%Precision	%Recall	%F-measure
Blog	75	60	66.67
	72.7	100	84.19
	100	100	100
	100	50	66.67
	100	50	66.67
Font	100	67	80.24
	80	100	88.89
	100	100	100
	83.33	100	90.91
	62.5	100	76.92
Password	100	100	100
	67	67	67
	75	100	85.71
	67	100	80.24
	100	75	85.71
Gravatar	100	75	85.71
	100	100	100
	71.42	83	76.78
	82.7	86	84.32
	66.87	67	66.93
Feed	100	67	80.24
	100	100	100
	85.7	100	92.30
	75	100	85.71
	71.4	83	76.76

Findings: Across the 25 screencasts, the approach reports a median of 23.08% of the transcript sentences as being use case steps. Given the high precision and recall of the approach in identifying use case steps, this low number of transcription sentences being required for describing these use case steps is a good indicator that our approach does also perform well in summarizing the relevant content of screencasts.

4.2 RQ2. How well do existing screencast ranking mechanisms perform in terms of relevancy of the mined speech content?

Approach: Existing search engines of online video portals, such as YouTube, use different criteria for ranking their result sets based on a user's search query. Most of the criteria are based on the similarity of the search keywords, relying on the textual description provided by the user (e.g., title, tags or video description), as well as the number of views or user ratings. However, none of these search engines analyzes and takes advantage of the actual speech content (picture or speech) in their ranking. The closest match is Google's STT, whose results (bag of words) are added to the textual description of a video. However, the actual meaning of the video in case of a screencast, i.e., the key steps to follow, are not considered.

It is not uncommon that videos with low quality content (e.g., not related to top of interest, or with an incorrect title) might be ranked first. In addition, criteria such as number of views might not always be a reliable indicator, since in some cases videos might go "viral", due to some publicity or a particular event, even if their content is not necessarily of high quality or relevant.

Future work is required to derive an optimal ranking algorithm for use case mining of screencasts. However, as a first step in this direction, we use the evaluation results of RQ1 and determine, per use case, optimal relevancy criteria for the 5 screencasts. We then compare the relevancy of the transcriptions to the following criteria: length of the screencast (measured by the number of words in the transcript), number of topic keywords appearing in a transcript, confidence score reported by the STT, reported feedback on the video (total number of likes and dislikes), and number of views of the video online. We define the optimal ranking as the one based on recall, with the idea that the most complete screencast in terms of use case steps is the most desirable one.

Given that, per use case, we only have recall values of 5 screencasts, our findings are preliminary. However, since we have 5 different use cases, we should be able to notice any trend. We use Spearman rank correlation, given the small sample size. Note that we cannot combine the 25 screencasts in one data set and calculate one correlation value, since some screencasts are in general more popular than others or require longer explanations, which would lead to incorrect correlation values. Table 6 contains the resulting correlation values.

Findings: The number of topic keywords in a transcript, and the STT confidence currently seem to be the best indicators of relevancy. Unsurprisingly, video length also has a high correlation, since the more one says in a screencast, the higher the probability to cover an additional use case step. Hence, this correlation is not that useful in practice. The fact that high confidence correlates with higher recall also might be by design, as our approach only keeps high confidence screencasts. On the other hand, if a topic model is built on a set of screencasts for a particular use case, any additional screencast for that use case might leverage the existing topic model such that a ranking based on number of topic keywords becomes feasible.

On the other hand, the number of views is not a reliable ranking, since correlations vary from very high, positive correlation to very high, negative correlations. This could confirm our remarks about viral videos or other sources of noise in this metric. Finally, feedback is also unreliable, giving equally extreme correlation values. Future work should try and further improve the current rankings.

Table 6. Correlations between transcript properties, for each use case.

Dataset	Recall-Length	Recall-Keywords	Recall-Confidence	Recall-Feedback	Recall-Views
Blog	0.81	0.89	0.79	0.65	0.63
Font	0.71	0.71	0.71	0	0
Password	0.67	0.89	0.89	-0.11	-0.78
Gravatar	0.60	0.70	0.70	-0.87	-0.40
Feed	0.34	0.22	0.45	-0.67	-0.67

5. Threats to validity

This research is taking a step towards making relevant content from screencast tutorials, created by the crowd, become an integrated part of existing software documentation. Our case studies illustrate that our methodology can deliver on these objectives. However, the data set and methodology are very different from existing approaches, leading to some challenges that might affect our reported results.

External Validity. In this research, a large, but still limited number of documents related to five use case scenarios of WordPress (a mature and widely used CMS) are analyzed. Therefore, to be able to generalize this approach and prove its applicability in different domains, it should be applied on larger datasets extracted from different domains and for extracting different types of documentation. While analyzing a larger number of speech files would increase the number of useful transcriptions with sufficiently high confidence score in the STT output, and therefore improve recall and precision for the approach, there are several factors that will affect the number of related videos being available online, such as: popularity and maturity of a software product, and the purpose of the video being analyzed (e.g., coding related, usage related, design related).

Internal Validity. In this work, we were able to gain high precision and recall values for the proposed mining methodology for crowd-based speech documentation. Nonetheless, various characteristics in the speech data output from STT tools can affect the results. STT tools typically output speech data with only limited punctuation, limiting the applicability of text analysis tools and APIs. Our approach mitigates some of this, by adopting a technique to segment text into sentences based on the pace of the voice in a speech document and pauses between words/sentences. Nonetheless, the resulting sentences are not always well-structured, nor do they map to normal full sentences in written documentation. To further mitigate this problem one could create domain-specific training data for speech text analysis that would allow us to build a language model during the sentence segmentation phase to improve the sentence structure of the transcripts.

Another potential threat to the internal validity of our approach is that, grammar used in spoken text differs significantly from the one in written text. This currently limits the use of written text dependency parsers for identifying sentence boundaries. This threat could be addressed by using spoken text dependency parsers as an alternative solution for sentence splitting.

A threat for the use of STT tools and the quality of produced transcripts is the dialect of the speaker in the speech files. In our dataset, we analyzed videos created by native and non-native English speakers and their English dialects (e.g., British, American, Indian, etc.). For our experiments, we used the US English dialect for analyzing the videos, potentially affecting the quality of our transcripts. Mitigating this thread would require to have an STT tool that supports multiple dialects in different languages.

In addition, low speech quality of the screencasts can reduce the quality of the transcripts in terms of their confidence score. We addressed in general the problem of low quality transcripts by filtering out those screencasts for which the STT tool reported a confidence score of less than 0.7. This filtered out 33 videos (see Table 2).

Reproducibility and Reliability. Our study has high reliability because the data is publicly available[13], we rely on publicly available third party STT tools and information extraction APIs, and our measures are proxies of those used in previous work [9]. Other researchers can replicate and expand upon our results.

6. Related Work

Over the last decade, there has been an ongoing trend in the mining software repository community to extract and analyze crowdsourced and crowd-based artifacts to support existing software artifacts. Existing research has explored the possibilities of leveraging such crowd-based documents using a variety of Information Extraction methods to analyze or extract tacit knowledge shared by developers through social media [2].

While most of this research has focused on StackOverflow, GitHub and blog posts (e.g., [2], [12], [13], [14]), the analysis of multimedia content to support developers (e.g., [3], [4], [5], [14]) is gaining momentum. The motivation behind analysis of such multimedia content is that software developers and users share frequently their domain knowledge through tutorial videos [3]. In the following, we provide an overview of work closely to our research.

6.1 Linking Crowd-based Documents

Jiau and Yang [15], measured the inequality of crowdsourced API documents in StackOverflow and found that a larger proportion of existing documents addresses a smaller portion of topics. They proposed a method of recovering traceability links for reusing existing documentations to lower this inequality. They claim that their proposed method improves documentation coverage by 400%. Barzilay *et al.* in [16], explored the design and characteristics of StackOverflow and developed a code search tool, *Example Overflow*, on top of StackOverflow to extract high quality code examples. As part of their empirical analysis, they studied the type of questions posted on StackOverflow and to what extent these questions could be answered by their approach. Subramanian *et al.* [17] proposed a method for linking code examples on StackOverflow to API documentations. Based on the proposed method, they implemented a tool, *Baker*, that links code snippets to Java classes and methods or JavaScript functions, with an observed precision of 97%. Bao *et al.* [14] proposed a method of tracking user activities and developed a tool, *ActivitySpace,* to support inter-application information needs of software developers. The tool reduces the efforts of developers for locating documents and recalling their history activities in daily work. In contrast to all this existing research is that our work focuses on analyzing multimedia crowd-based documentation.

6.2 Analyzing Crowd-based Documents

Many exploratory studies have been conducted to analyze the characteristics of written crowd-based documents on the Web (e.g., [12], [18], [19]). Parnin and Treude [12] focused in their work on the possibility of crowd-based documents replacing traditional software documentation, by measuring to what extent blog posts cover methods of a particular API. They observed that social media posts can cover 87.9% of the API methods and therefore could potentially replace documentation. Nasehi *et al.* [13] analyzed code examples of StackOverflow, that are voted as being good code examples to find the characteristics that, if applied, can improve developing and evolving API documentations. Another research on StackOverflow conducted by Campbell *et al.* [18], used LDA to find topics in PHP and Python documentation that do not overlap

[13] http://1drv.ms/1SlGviQ

with the topics on StackOverflow. Their results show that many topics, not covered by traditional project documentation, are related to tutorial documentations. Pham *et al.* [19] investigate in their research the possibility of extracting a common testing culture to tackle testing challenges. They conducted a survey among GitHub users to identify challenges arising from using social coding sites such as GitHub, and their impact on testing practices.

Recently, MacLeod *et al.* [3] conducted a study on the use of screencasts in sharing programming knowledge. They found that developers use screencasts to share information such as: 1) how to customize a program, 2) the challenges they encountered and their development experiences, 3) solutions to problems, 4) how to apply design patterns, and 5) their programming language knowledge. They also surveyed 10 developers who created screencasts on YouTube about their motivation for creating these screencasts. Their results show that most screencast creators are creating these videos to promote themselves and gain reputation by helping others. Bao *et al.* [4], [5] developed a video scraping tool, *scvRipper*, to automatically extract time-series Human Computer Interaction data from screen-captured videos. They used their computer-vision based technique to detect and extract actions of a developer by employing key point based template matching to differentiate key points from each other.

Common to these reviewed approaches for linking and analyzing crowd-based documents is that they are mostly concerned with analyzing textual or visual documentation, while our approach focuses on the transcription and analysis of speech content, not only to supplement existing documentation but also to improve the identification of relevant documents (videos) based on their content.

7. Conclusions and Future Work

This work describes an approach for mining the speech component of YouTube videos as one common type of crowd-based software documentation. We introduce a methodology that transcribes and analyzes the transcribed text using various Information Extraction techniques, and present a proof of concept implementation of our methodology. A case study was conducted on WordPress tutorial videos posted on YouTube to illustrate that it is possible to extract useful software documentation, in our case, use case scenarios. We also showed that the use of the analyzed video content can potentially provide an improved ranking of crowd-based documentation resources with a speech component.

As part of our future work, we plan to evaluate different statistical models to extract abstract topics and improve our sentence segmentation using other techniques, such as language models.

8. REFERENCES

[1] Turk, D., France, R., and Rumpe, B., (2002), Limitations of Agile Software Processes, Proceedings of the Third International Conferenceon extreme Programming and Agile Processes in Software Engineering, p. 43-46.

[2] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow," Georgia Institute of Technology, Tech. Rep., 2012.

[3] L. MacLeod, “Code, Camera, Action!: How Software Developers Document and Share Program Knowledge Using YouTube,” in *2015 IEEE 23rd International Conference on Program Comprehension*, 2015, p. 11.

[4] L. Bao, J. Li, Z. Xing, X. Wang and B. Zhou, "Reverse engineering time-series interaction data from screen-captured videos", Proc. 22nd IEEE International Conference on Software Analysis, Evolution and Reengineering, pp. 399-408.

[5] L. Bao, J. Li, Z. Xing, and X. Wang, “Extracting and analyzing time-series HCI data from screen-captured task videos,” *Empir. Softw. Eng.*, 2016, pp. 1–41.

[6] A. Pappu and A. Stent, “Automatic Formatted Transcripts for Videos,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2015, pp. 2514–2518.

[7] T. Oba, T. Hori, and A. Nakamura, "Sentence boundary detection using sequential dependency analysis combined with crf-based chunking," vol. 3, 2006, pp. 1153-1156.

[8] T. Oba , T. Hori and A. Nakamura, "Improved sequential dependency analysis integrating labeling-based sentence boundary detection", IEICE Trans. Inf. Syst., vol. E93-D, no. 5, 2010, pp. 1272-1281.

[9] S. Takahashi and T. Morimoto, “N-gram language model based on multi-word expressions in web documents for speech recognition and closed-captioning,” *Proc. of the Int. Conf. Asian Lang. Process. (IALP),* 2012.

[10] Michael W. Berry, Ed., *Survey of Text Mining*. Springer New York, 2004.

[11] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *J. Mach. Learn. Res.*, Mar. 2003, vol. 3, pp. 993–1022.

[12] C. Parnin and C. Treude, "Measuring API Documentation on the Web," in Proceedings of Web2SE 2011, New York, NY, USA, 2011, pp. 25-30.

[13] S.M. Nasehi, "What Makes a Good Code Example? A Study of Programming Q&A in StackOverflow", Proc. 28th IEEE Int Conf. Software Maintenance, 2012, pp. 25-34.

[14] L. Bao, Z. Xing, X. Wang, and B. Zhou, “Tracking and Analyzing Cross-Cutting Activities in Developers’ Daily Work.”, In: Proceedings of the 30th IEEE/ACM international conference on automated software engineering.

[15] H. C. Jiau and F.-P. Yang, “Facing up to the inequality of crowdsourced API documentation,” *ACM SIGSOFT Softw. Eng. Notes*, 2012, vol. 37, no. 1, pp. 1-9.

[16] O. Barzilay, C. Treude, and A. Zagalsky, Facilitating Crowd Sourced Software Engineering via Stack Overflow. New York: Springer, 2013, pp. 297-316.

[17] S. Subramanian, L. Inozemtseva, and R. Holmes, “Live API documentation,” in *Proceedings of the 36th International Conference on Software Engineering - ICSE*, 2014, pp. 643-652.

[18] J.-C. Campbell, Chenlei Zhang, Zhen Xu, A. Hindle, and J. Miller, “Deficient documentation detection a methodology to locate deficient project documentation using topic analysis,” *10th IEEE Work. Conf. Min. Softw. Repos. (MSR)*, 2013, pp. 57-60.

[19] R. Pham, L. Singer, O. Liskin, F. Figueira Filho, and K. Schneider, “Creating a shared understanding of testing culture on a social coding site,” *35th Int. Conf. Softw. Eng. (ICSE)*, 2013, pp. 112-121.