Automatic Tag Recommendation for Software Development Video Tutorials

Esteban Parra
Florida State University
Tallahassee, Florida
parrarod@cs.fsu.edu

Javier Escobar-Avila
Florida State University
Tallahassee, Florida
escobara@cs.fsu.edu

Sonia Haiduc
Florida State University
Tallahassee, Florida
shaiduc@cs.fsu.edu

ABSTRACT

Software development video tutorials are emerging as a new resource for developers to support their information needs. However, when trying to find the right video to watch for a task at hand, developers have little information at their disposal to quickly decide if they found the right video or not. This can lead to missing the best tutorials or wasting time watching irrelevant ones.

Other external sources of information for developers, such as StackOverflow, have benefited from the existence of informative tags, which help developers to quickly gauge the relevance of posts and find related ones. We argue that the same is valid also for videos and propose the first set of approaches to automatically generate tags describing the contents of software development video tutorials. We investigate seven tagging approaches for this purpose, some using information retrieval techniques and leveraging only the information in the videos, others relying on external sources of information, such as StackOverflow, as well as two out-of-the-box commercial video tagging approaches. We evaluated 19 different configurations of these tagging approaches and the results of a user study showed that some of the information retrieval-based approaches performed the best and were able to recommend tags that developers consider relevant for describing programming videos.

CCS CONCEPTS

• **Software and its engineering** → *Software libraries and repositories*; *Documentation*; • **Information systems** → *Summarization*; *Multimedia and multimodal retrieval*;

KEYWORDS

Video tutorials, automatic tagging, software engineering, information retrieval

ACM Reference Format:
Esteban Parra, Javier Escobar-Avila, and Sonia Haiduc. 2018. Automatic Tag Recommendation for Software Development Video Tutorials. In *ICPC '18: ICPC '18: 26th IEEE/ACM International Confernece on Program Comprehension , May 27–28, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3196321.3196351

ICPC '18, May 27–28, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5714-2/18/05…$15.00
https://doi.org/10.1145/3196321.3196351

1 INTRODUCTION

Nowadays, software developers often rely on online resources (e.g., Q&A sites, online API documentation, online tutorials, etc.) to stay up to date and learn new information relevant to their daily tasks. Software engineering video tutorials are one such resource. These videos usually deliver introductory or in-depth information regarding software engineering topics, such as explaining programming language syntax, step-by-step instructions for designing and implementing algorithms or data structures, demonstrating the use of APIs or design patterns, configuring programming environments, solving errors, and so on [19]. While videos are becoming more and more popular on video-sharing websites such as YouTube [42], software development video tutorials are still an underused resource due to several existing limitations which can hinder their adoption [27]. One such limitation is the difficulty in quickly determining if a video contains the answer to a particular information need somewhere in its contents without having to watch the video. Being able to assess the contents of a video quickly is particularly useful given the increasing number of software development videos available, and therefore the need to sift through many videos that may potentially contain the needed information.

While videos hosted on platforms like YouTube usually have a title and description which provide a general overview of the video's contents, they are not always the best solution for getting a quick, concise overview of the most important concepts in the video. The title usually does not capture all the key topics discussed in the video, while the description can sometimes be irrelevant (e.g., listing the social media channels of the author), or too long to read, given that people spend less than a second analyzing an online resource to determine its relevance [13].

Tagging is the labeling of a particular resource using keywords that are relevant for describing the content of that resource. Tagging has been successfully used to classify and label a wide variety of online resources such as news, website bookmarks, scholarly references, pictures, reusable software libraries, text-based tutorials, and posts in developer Q&A sites, such as StackOverflow[1]. We believe that similar to these other online resources that have benefited from tagging, tags for software development videos could help developers in various ways [12]. First, they could provide them with the list of the most representative concepts discussed in the video in a clear, concise, and effective way, which could help them better gauge the relevance of a particular video in a short amount of time. Second, tags could support navigation and discovery of related content to a video of interest, by looking for other videos having the same or similar tags.

[1] https://stackoverflow.com/

We therefore propose augmenting the available user metadata of software development videos with relevant tags that capture the main concepts discussed in the video. In particular, in this paper we focus on videos hosted on YouTube, as it is the most popular video-sharing platform at the moment. We propose and evaluate a set of 19 different configurations across seven approaches for automatically tagging software development videos. The approaches we investigate leverage a variety of information retrieval and natural language processing techniques in order to extract the most relevant keywords for a video from the pool of words in its title, description, audio transcript, as well as from external resources related to the content of the video, such as StackOverflow posts and tags. We also investigate two out-of-the-box, closed-source commercial video tagging approaches and evaluate how well they perform on software development videos.

To evaluate the quality of the tags produced by the automatic tagging approaches, we performed a user study where 57 participants provided manual tags for a set of 75 software development videos about Java programming extracted from YouTube. The manual tags were then compared with the automatically generated ones to determine the performance of the 19 approaches. To better understand the process of manual vs. automatic tag generation, we also investigated the provenance of manual and automatic tags to determine if the users and the tagging approaches chose terms from the same source or not.

Our results show that the tags recommended by one of the information retrieval tagging configurations, based on BM25F, performs the best, reaching a high level of agreement with the tags provided by the study participants. The study also revealed that the approaches leveraging external sources of information (i.e., StackOverflow) perform poorly when tagging software development video tutorials, indicating that knowledge transfer between different developer documentation platforms is a challenging problem.

The rest of the paper is structured as follows: section 2 introduces the process and the techniques we used to recommend tags for software development videos, section 3 presents the user study we performed for evaluating the automatic tagging approaches explored, while section 4 summarizes the results of the study. We discuss threats to validity in section 5, section 6 presents related work, and section 7 concludes the paper and discusses future work.

2 AUTOMATIC TAGGING FOR SOFTWARE ENGINEERING VIDEOS

In this section, we describe the approaches we consider to automatically generate tags for software development videos. We divide them into three categories: information retrieval-based, StackOverflow-based, and commercial, closed-source approaches. Many of the approaches make use of a video's metadata (i.e., title, description, and audio transcripts) in a direct or indirect way to recommend tags. Some approaches can only produce tags representing words that appear in the video metadata; we call these approaches *extractive*. Other approaches, however, could return tags that do not appear anywhere in the video. This is due to either the fact that they abstract the topics discussed in the video on a higher-level, or because they use external information such as StackOverflow to produce the tags. We call these types of approaches *abstractive*.

2.1 Information Retrieval-based Approaches

Approaches based on information retrieval (IR) have been successfully used for term-based summarization and tagging of natural language documents and software artifacts [9, 14]. Therefore, we believe this type of techniques may be useful for tagging other types of software artifacts, such as software development video tutorials. We propose a few IR approaches for video tagging, which we briefly describe below. For more information about them, we refer the reader to [6, 7].

Term Frequency - Inverse Document Frequency (TF-IDF) is a popular, yet simple *extractive* approach to identify important keywords for a document (i.e., video) by rewarding terms that frequently appear in the document but are not found across many other documents in a corpus [15]. Term frequency (TF) is the number of times the term appears in a document, whereas, Inverse Document Frequency (IDF) is the logarithm of the inverse of the number of documents in which the term appears in the corpus. The TF-IDF score of a term is the product of its TF and IDF values. For our purposes, we use the Terrier [18] implementation of TF-IDF to extract the terms in a video's metadata (i.e, title, description, and transcript) with the highest TF-IDF score.

Latent Dirichlet Allocation (LDA) is an *abstractive* approach that determines *topics* from a set of documents via a statistical model that represents each document as a mixture of topics, with each topic having a particular probability of generating a term in the document. In our experiments, we use LDA-GA [22] to find a near optimal parameter configuration for LDA using a genetic algorithm.

BM25F is a semi-structured ranking function based on a probabilistic retrieval framework [32] where terms can be given different weights (i.e., importance) depending on the field in which they appear in a document (i.e., title, description, or transcript in our case). We denote a particular choice of weights for the three fields as $BM25F(T, D, Tr)$, where T, D, and Tr are the weight factors given to terms from the title, description, and transcripts, respectively.

BM25F computes the final weight of a term as its accumulated weight over all the fields in a document, as given by equation (1), where $freq^d_{t,c}$ is the term frequency of the term t in field c of document d, $boost_c$ is the boost factor or importance of field c, l_c is the length of field c in the document (i.e., the number of terms in the field), and avl_c is the average length of field c across all documents in the corpus.

$$weight(t,d) = \sum_c \frac{freq^d_{t,c} \cdot boost_c}{(1-b_c) + b_c \cdot \frac{l_c}{avl_c}} \quad (1)$$

The weight of a term is adjusted by a non-linear saturation function $\frac{weight(t,d)}{k_1 + weight(t,d)}$ to reduce the effect of term frequency on the final score. This model requires two parameters, k_1 and b_c, where k_1 is a free parameter usually set to 2 and $b_c \in [0, 1]$ is usually set to 0.75 [32]. We used the default values as we found they produce good results in our experiments.

In BM25F, field boosting factors ($boost_c$ in equation (1)) are used to modify the importance of each field in the ranking function. The choice of boosting factors can have a major impact on the weight of terms [32]. We evaluated 12 different BM25F configurations,

using various degrees of importance for the different fields. We used the Terrier IR engine to compute the score of each term and then extracted the terms in a video with the highest BM25F score as its tags. These BM25F-based approaches are *extractive* in nature.

2.2 StackOverflow-based Approaches

StackOverflow (SO) is a Q&A site for programming topics that employs tags to categorize questions. Every post in SO is tagged by its creator using up to five tags from a set of more than 45,000 tags curated by the community. We propose a new *abstractive* tagging approach for software development videos which uses IR techniques and leverages the tags contained in SO, considering them as potential tags for video tutorials. We also adapt the state-of-the-art tag recommendation approach for large-scale software information sites, called TagMulRec [44] and use it to recommend tags for software development videos. We describe both approaches below.

IR StackOverflow-based Tagging: We propose a new approach that employs IR techniques and uses SO tags to label software development videos. The approach works as follows: 1) using the SO dump[2], collect all the questions related to particular topics of interest. In our case, we were interested in labeling Java-related videos, so we focused only on SO questions having the "java" tag; 2) collect all the tags appearing in the set of relevant questions and compute their frequency (i.e., how many questions are labeled using each tag). We removed the tags whose frequency was below the first quartile, to account for specialized tags that appear in very few SO questions; 3) create a single text document for each considered SO tag. This text document contains the text of all the questions and accepted answers labeled with that tag; 4) index each text document using Apache Lucene[3]; and 5) use the metadata (i.e., title, description, and transcript) of a particular video as a query and retrieve the most relevant documents (i.e., tags) to it. The tags represented by these documents will be recommended as tags for the video considered as a query.

Lucene can be parametrized with the similarity measure used to compare the query and an indexed document. We used two different similarity measures based on language models, namely LMDirichletSimilarity and LMJelinekMercerSimilarity. These resulted, therefore, in two different tagging configurations based on using SO data.

TagMulRec [44] is the state-of-the-art tag recommendation approach for large-scale software information sites (e.g., StackOverflow). It is a tag propagation approach that uses existing tagged Q&A posts to automatically recommend tags for new Q&A posts. Given a new Q&A post, TagMulRec first locates the top n posts that are the most semantically similar to it and uses their respective tags as an initial set of candidate tags. For each candidate tag a relevance score is computed by aggregating the semantic similarities of all the posts in the top n most similar posts that are tagged with it. Then the tags are ordered based on this score and those with the highest relevance are assigned as tags for the new post.

We adapted TagMulRec to tag programming video tutorials instead of SO posts as follows: we first created a document representing each video, by concatenating all the video metadata (i.e., the title, description, and transcript); then, we used TagMulRec trained with the full SO data dump [2] to retrieve the most relevant SO tags for each video document (instead of a SO post).

2.3 Closed-Source Commercial Approaches

We also investigate two existing, closed-source commercial approaches for tagging videos, described below.

Semantic Fingerprint (Cortical.io API): Semantic fingerprinting [38], as implemented in the Cortical.io[4] API , is a machine learning approach that proposes a new semantic representation for terms and documents. Using an existing trained model[5], Cortical.io builds a semantic fingerprint for an entire text document and for every term in such document. The fingerprints of each term are then compared with the fingerprint of the document to compute their overlap. The overall importance of a term is given by its overlap with the documents' fingerprint, weighted by the frequency of the term in the document. We choose the top words identified by Cortical.io for a video as its recommended tags based on this *extractive* approach.

Google Cloud Video Intelligence API[6], referred to as GVI through the remainder of the paper, is Google's implementation of a Video2Text approach [2]. GVI is a large scale deep learning system that uses a pre-trained neural network model to analyze the audiovisual signal of a video and automatically extract key entities from the video (e.g., dog, flower, car), and when they occur within the video. We uploaded the videos in our dataset to Google Cloud storage and processed them with GVI to obtain all the labels for the video, and their confidence score. For each video, we sorted the results by confidence level in descending order and selected the top labels as tags. GVI is an *abstractive* approach by construction.

3 STUDY DESIGN

We performed an empirical study with the *goal* of evaluating the automatic tagging approaches described in Section 2 by comparing their video tag recommendations with tags manually created by software developers. In this study, we particularly focus on tagging Java programming videos hosted on YouTube. The *quality focus* of the study concerns the quality and provenance of the automatically generated video tags, as compared to manual tags. The *perspective* is that of researchers interested in understanding the extent to which IR-based, SO-based, and out-of-the-box commercial approaches can be used to generate tags for software development videos and the factors that can impact their performance.

In the context of our study, we specifically aim to answer the following research questions:

RQ1 - Quality: ***What is the quality of the tags produced by the automatic tagging approaches for software development videos?*** This is the main research question of our study and aims at quantifying the performance of the studied automatic tagging approaches in recommending tags that resemble those produced by software developers. We are particularly interested in determining the best performing approaches, as well as studying some of the

[2]We used the one published as of March 2017
[3]https://lucene.apache.org/core/
[4]http://www.cortical.io
[5]The trained model is an internal implementation of the API. Details about the implementation are not disclosed.
[6]https://cloud.google.com/video-intelligence/

factors that can impact their performance in producing quality tags. In particular, we study the following subquestions:

- *RQ1.1 Which automatic video tagging approach produces the best tags?* In this subquestion we were interested in determining the best performing approach among the ones studied and how well it can approximate the tags given by developers.
- *RQ1.2 Do singularization and stemming impact the performance of the automatic video tagging approaches?* Morphological transformations of words can have an effect on the performance of Natural Language Processing and Information Retrieval systems [3]. Therefore, we study the effect of *singularization* and *stemming* on the approaches for tagging software development videos. Singularization is the transformation of a word in the plural form to a singular form (e.g., classes → class, arrays → array), whereas stemming is the modification of an inflected word to its word stem or root form (e.g., programming → program, properties; property → properti). We used the jBoss[7] Inflector for singularization and the Stanford-NLP[8] Porter Stemmer for stemming.

RQ2:Where do the tags that developers perceive to be relevant for software development videos come from?

This research question aims at analyzing the origin of tags provided by developers and compare it to that of the automatic tags. This information could be used as guidelines to produce better tags for software development videos by future approaches.

The following subsections describe the context of our study and the methodology used to address our research questions.

3.1 Participants

In total, 57 participants took part in our user study, all of whom had experience with software engineering tasks and at least six months programming experience in Java. The participants were 15 Junior and Senior undergraduate students, 22 Master students, 17 Ph.D. students, two professional developers, and one faculty member. The participants were recruited across several higher education institutions in the USA and South America.

3.2 Video Dataset

In this study, our focus is on tagging Java-related videos hosted on YouTube. YouTube provides a REST API which allows developers to search for videos using keywords and extract detailed information about a video given its unique identifier. To obtain a comprehensive dataset of YouTube videos about programming in Java, we used the YouTube API to search for the query "Java programming" and retrieve the videos that were obtained in response to it and were uploaded on YouTube between January 2010 and June 2016. We also restricted our search to videos in the English language. The API also provides the option to retrieve the Freebase topic of a video, which is a field used to link the video to a single high-level topic within a collaborative knowledge base[9]. We use the Freebase topics associated with the videos to filter out irrelevant results. In particular, we collected and manually reviewed all the Freebase topics associated with our initial set of videos, obtaining a list of 69 Freebase topics relevant to Java programming. We then only kept videos associated with the topics on this Freebase topics list, resulting in almost 53,000 videos. We extracted the title and description for each of these videos, but the YouTube API does not offer an option to download the transcripts of a video.

Since the audio transcript of a video is a crucial component in generating tags using our proposed approaches, we needed to extract also the audio transcript of videos when one was available. Based on the method used for their creation, transcripts of software development videos can be either manual or automatic [8]. Manual transcripts are created when a person manually transcribes the content of the video into a text file, whereas, automatic transcripts are created via Automatic Speech Recognition (ASR) software that transforms an audio signal into text.

We used the YouTube API to determine if a video has manual transcripts provided by the creator of the video. From the initial 53,000 videos, we identified 1,045 with manual transcripts, while the rest either had automatic transcripts or no transcripts at all. The YouTube API does not allow downloading transcripts, so we used youtube-dl[10] to download the available manual and automatic transcripts of the videos. Out of the initial set, we were able to extract only 846 manual transcripts (i.e., provided by developers) and 31,539 automatic transcripts (generated by YouTube's ASR) in English.

After the analysis of a subset of the extracted transcripts, we detected that the automatic transcripts created by YouTube via ASR are very noisy and contain numerous mistakes. Since automatically extracting correct transcripts is an entirely different problem, we chose to focus for now on video tutorials for which manual transcripts were available. However, the proposed approaches can be applied on automatic transcripts as well, though their output will be dependent on the quality of those transcripts.

At a closer look we also detected that some manual transcripts provided by the creator of the video did not correspond to what was spoken. For example, some authors used the transcript to publicize their web pages or include contact information instead of transcribing the audio of the video. We refer to this kind of transcripts as "low-quality" manual transcripts. We used heuristics to filter out the videos that had low-quality manual transcripts. First, we collected the duration of every video in the 846 that had manual transcripts associated with them. Then, we computed the number of words in their manual transcripts per second of video duration. Next, by analyzing the distribution of the word-per-second ratio, we observed that the transcripts that had less than one word per second were usually incorrect, corresponding mostly to ads or other text that did not reflect what was being talked about in the video. We therefore removed all videos whose transcripts had a word-per-second ratio lower than one. As a result of this process, we were left with 533 videos with good manual transcripts according to our heuristic. We want to underline the fact that, even though for feasibility purposes we further reduce this number of videos to 75 in our user study, some of our automatic tagging approaches make use of metadata and transcript information from all the 533 videos. For example, LDA uses this entire set to extract topics from the whole video collection that are then used to generate tags for particular videos.

[7] www.jboss.org/
[8] https://nlp.stanford.edu/software/
[9] https://goo.gl/eJAvZn
[10] https://rg3.github.io/youtube-dl/

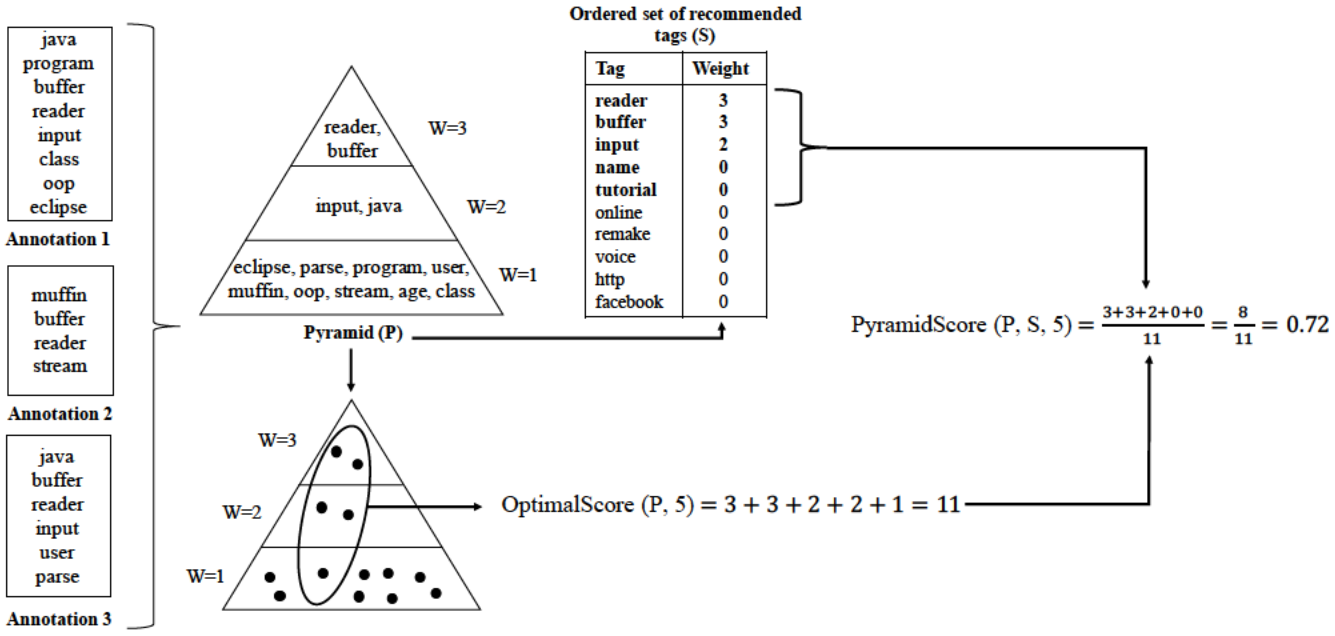

Figure 1: Pyramid creation and pyramid score computation

3.3 User Tags and the Ground Truth

The length of the 533 videos in our dataset ranged from 2 to over 20 minutes, with a median duration of 6.9 minutes. To ensure the evaluation of more videos while limiting the fatigue effect on our study participants, we only considered videos with a length of up to 7 minutes. This resulted in a reduced set of 315 videos. Based on the number of participants recruited for our user study, we randomly generated a sample of 75 videos that we use in our study.

We used an online questionnaire for data collection. The questionnaire was composed of two sections. Section 1 was used to collect demographic information regarding the participants (e.g., programming experience, known programming languages, and education level), whereas section 2 asked the participants to watch a set of three or five randomly selected videos, one at a time, and then provide manual tags to describe each video. In particular, 40 participants watched three videos, whereas the other 17 participants watched five videos. In addition to the videos themselves, the questionnaire also showed the title and description of each video to the participants, to mimic what they would see on YouTube.

For the tagging of each video in the questionnaire, the participants were asked to watch the video from beginning to end and to provide tags that describe the video, in the order of their relevance. In the context of our study, a relevant tag is one that effectively describes what the video – or a part of it – is about. We did not set any limitations on the number of tags a participant could use to describe the video because we were interested in collecting a comprehensive set of tags.

To diminish the learning effect, we ensured that each participant watched videos on different topics, as well as videos created by various authors. We also distributed the videos so that the participants could complete the study within 25 to 45 minutes since research has shown that significant error rates linked with cognitive fatigue appear after engaging in a task for longer than 60 minutes [24].

After the 57 participants completed the study, we obtained the set of manual tags for the videos. We refer to the set of tags assigned by a user to a video as an *annotation*. In particular, we obtained 56 videos with three annotations, 14 videos with two annotations, and five videos with only one annotation.

After manually inspecting the annotations, we identified that 28.8% of the tags provided by the users were composite tags, which are tags composed of one or more words (e.g., exception-handling, arrayvsarraylist). We processed the tags provided by developers to extract single-term tags to do a fair comparison with the single-term tags produced by the evaluated automatic approaches. In particular, we split all composite tags by special characters and spaces, keeping unique single-term tags and removing the original composite tags. Also, we manually inspected the list of single-term tags to split any composite tags where special character separators were not used (e.g., niopackage → nio, package).

We used the resulting single-term tags for the evaluation. Each annotation had between 3 and 75 single-term tags, with an average of 9.55 single-term tags per video.

The Pyramid Method. The Pyramid Method is a summarization assessment methodology from the document summarization field [21] that has been successfully employed in Software Engineering for similar tasks to ours [14, 29]. We chose to use the pyramid method as it provides a reliable assessment of content selection quality in tagging and summarization and it allows building the ground truth from multiple annotations for the same video [21, 29]. In our study, this translates to each video having an associated pyramid-shaped, multi-tier partition of the ground truth tags provided by the annotators, where each tier in the pyramid contains only the tags with the same weight. The weight of a tag in the pyramid is the number of participants that used that tag in their

Table 1: Average pyramid scores for term-based summarization approaches

Approach	Unprocessed		Singular		Stem	
	5	10	5	10	5	10
TF-IDF	0.51	0.49	0.53	0.52	0.50	0.49
LDA	0.06	0.05	0.11	0.08	0.11	0.08
BM25F(0,0,1)	0.56	0.56	0.61	0.59	0.60	0.57
BM25F(0,1,0)	0.41	0.38	0.43	0.41	0.43	0.40
BM25F(1,0,0)	0.51	0.40	0.54	0.41	0.54	0.41
BM25F(1,1,0)	0.47	0.40	0.50	0.43	0.49	0.43
BM25F(1,1,1)	0.63	0.59	0.65	0.62	0.65	0.61
BM25F(1,1,2)	0.60	0.59	0.65	0.61	0.64	0.60
BM25F(1,2,0)	0.45	0.40	0.48	0.43	0.47	0.42
BM25F(1,2,4)	0.60	0.58	0.64	0.61	0.64	0.59
BM25F(1,4,2)	0.62	0.58	0.63	0.61	0.64	0.60
BM25F(2,1,0)	0.50	0.43	0.52	0.45	0.51	0.45
BM25F(2,2,1)	0.63	0.57	0.65	0.61	0.65	0.60
BM25F(4,2,1)	**0.64**	**0.59**	**0.66**	**0.62**	**0.66**	**0.61**
Cortical.io	0.56	0.48	0.56	0.51	0.43	0.37
SO-based tagging(LMDirichlet)	0.15	0.12	0.12	0.06	0.06	0.06
SO-based tagging(LMJelinek)	0.15	0.12	0.12	0.06	0.06	0.06
TagMulRec	0.35	0.30	0.35	0.32	0.35	0.32
GVI	9.30E-4	7.80E-4	9.30E-4	7.80E-4	8.70E-4	7.50E-4

annotation. The higher the tier on which a tag is found in the pyramid, the more people used that tag in their annotations. The bottom tier of the pyramid contains all the tags that were recommended by only one of the annotators of the video, while the top tier contains the tags that were recommended by the most annotators. Figure 1 exemplifies the creation of a pyramid (**P**) and the assignment of weights given the sets of tags (i.e., annotations) provided by three developers. Since pyramids built from a single annotation are known to be unreliable [21], we removed five videos from our evaluation, which received only one annotation each. The remaining 70 videos had two or three annotations each, which allowed the construction of valid pyramids.

Given a pyramid, we obtain the informativeness of a set of automatically recommended tags T by computing its pyramid score, which is the ratio of the sum of the weights of the tags in T to the sum of the weights of an optimal set of tags with the same number of tags as T. An optimal set of tags would contain all the tags from the higher tiers before including tags from lower tiers. By design, recommended tags that do not appear in the ground truth pyramid are assigned a weight of zero.

The pyramid score of a set of tags is mathematically expressed in Equation (2), where x is the number of tags in the set; P is a pyramid; S is a set of ordered tags; $weight(P, t)$ is the weight of tag t in the pyramid P; S_i is the i-th tag in the set S from top to bottom; and P_j is the j-th tag in the pyramid from top to bottom. A pyramid score ranges from 0 to 1, with higher scores indicating better sets of tags. Figure 1 presents a concrete example based on a recommendation of five tags.

$$PyramidScore(P, S, x) = \frac{\sum_{i=1}^{x} weight(P, S_i)}{\sum_{j=1}^{x} weight(P, P_j)} \quad (2)$$

4 RESULTS

We make all of our data and results used in this paper available in our replication package[11].

4.1 RQ1.1 - Best tagging approach

We initially consider recommendations of 10 tags for each of the approaches, in order to resemble the average number of tags that developers used to tag the videos in our study (i.e., 9.55 tags). For each tagging approach, we computed the average pyramid score of the recommended sets of tags across the 70 videos with two or three annotations. The average pyramid scores for 10 tags for each approach are presented in the column Unprocessed-10 in Table 1.

The tags produced by GVI are the least informative with average pyramid scores under 0.01, followed by LDA and the two variations of the SO-based tagging technique, all with scores lower than 0.20. The TagMulRec-based approach performs better than other abstractive approaches with an average pyramid score of 0.30. However, none of the abstractive approaches can surpass the extractive ones.

The best performing approaches are BM25F(4,2,1), BM25F(1,1,2), and BM25F(1,1,1) which all achieve an average pyramid score of 0.59 when recommending 10 tags. In other words, the tags obtained using these extractive approaches contain on average up to 59% of the information considered to be relevant by developers for describing the videos, whereas the tags provided by abstractive approaches only contain up to 35% such information. Since all the worst performing approaches are abstractive, we inspected the tags recommended by these approaches and noticed that they capture high-level information that is not as useful to quickly assess the content of a software development video.

[11] https://www.cs.fsu.edu/~serene/parra-icpc2018

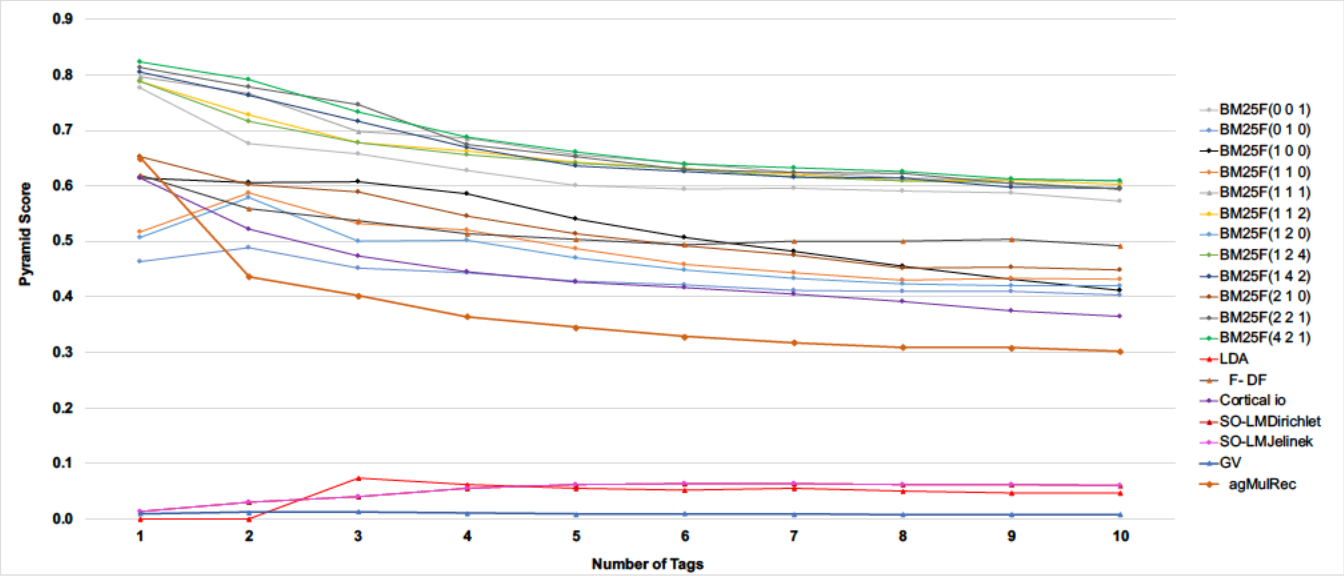

Figure 2: Average pyramid scores with variable number of tags

Our results indicate that extractive tagging approaches are better suited for tagging software development videos. Moreover, configurations of BM25F that give non-zero weight to terms in the transcript are more informative than configurations leveraging only terms from the title and description. Moreover, considering only the transcript (i.e., BM25F(0,0,1)) leads to an average pyramid score of 0.56, whereas configurations considering only terms from title and description achieve average pyramid scores of up to 0.43 (i.e., for BM25(2,1,0)), indicating that the transcript is an essential source of information for producing informative tags. This differences in score between the BM25F approach using only the transcript and the variants using only the title and description are statistically significant, as confirmed using Wilcoxon rank sum tests with 95% confidence interval (p-values<0.05 and a medium Cliff's delta effect size).

Number of Recommended Tags

The number of tags recommended can affect the effectiveness of the tagging approaches. Moreover, there is no existing work investigating how many tags should be used to label a software development video.

Initially, we choose to recommend 10 tags to resemble the average number of tags that developers used when tagging software development videos in our study – an average of 9.55 tags per video. However, there was a high variation in the number of tags assigned by individual developers in our study. Specifically, some developers used three tags, whereas one developer used up to 75 tags to annotate a single software development video.

We use the variation of pyramid score when recommending a different number of tags to determine the number of tags that we should recommend when using automatic approaches. In particular, we computed the pyramid scores for all the approaches when varying the number of tags from one to ten to investigate whether changing the number of recommended tags has an impact on the performance of the approaches. The average pyramid scores for all the approaches when recommending a varying number of tags are shown in Figure 2.

The results indicate that the overall pyramid score of the approaches decreases as we increase the number of tags recommended, with the exception of BM25F(1,0,0), SO-LMDirichlet, SO-LMJelinek, and LDA, which increase when recommending two or three tags but start decreasing when recommending four or more tags. This shows that the informativeness of the set of recommended tags is impacted not only by the number of tags, but also by the rank of the most relevant tags.

In addition to the overall tendency for the scores to decrease, we notice that the largest difference on the value of the pyramid score tends to occur between recommending one tag and recommending three tags, a 0.11 drop for BM25F(4,2,1) – the best performing approach. Also, the scores do not vary significantly after recommending five tags, indicating that the most informative tags are recommended first. This was confirmed after comparing the pyramid scores of 5 tags and 10 tags using Wilcoxon sum rank test (p-value = 0.794 after using Holm-Bonferroni correction). Due to space limitation, we only present a detailed overview of the variation of pyramid scores when recommending five and ten tags in Table 1.

One of the best performing approaches when recommending ten tags, namely BM25F(4,2,1), outperforms all the other approaches when recommending one to nine tags. In particular, the average pyramid score of the tags recommended by BM25F(4,2,1) increases from 0.59 when recommending ten tags to 0.64 when recommending only five tags, whereas the average pyramid scores of the tags recommended by other configurations do not surpass 0.63 when recommending five tags. Based on these results, we believe five to be a good number of tags for tagging software development videos, given also that five tags have been proven to work well for

other software development related documents (i.e., Q&A posts in StackOverflow). Moreover, the extractive tagging approaches exhibit comparable performance when recommending a various number of tags, and the informativeness of the automatic tags produced does not deteriorate quickly after recommending five tags.

4.2 RQ1.2 - Singularization and Stemming

Morphological transformations, particularly stemming, can increase the informativeness of automatically extracted tags by reducing the ambiguity of specific terms in the corpus, as well as resolving, to a degree, differences between the tags provided by users. Consider the case where participant A annotates a video with the following tags {column, importing, nullable, class}, and participant B annotates the same video with {imports, nullable, classes, row}. A pyramid built with the previous annotations will result in only one highly ranked tag – nullable, despite the fact that there are other tags that should be ranked higher because they provide the same semantic information (i.e., class and classes; imports and importing). After stemming, the annotations from A and B become {column, import, nullabl, class}, and {import, nullabl, class, row}, respectively. A pyramid built from the stemmed annotations will have the three most informative tags – nullabl, class, import – on the top level. In this example, stemming creates a pyramid that is semantically stronger at reflecting the concepts considered to be important by the annotators. Moreover, if the video does not use the words imports or importing but frequently uses the word imported, the stemming of the documents allows for the tag import to be recommended by extractive approaches.

To measure the impact of using morphological transformations, we created a singularized corpus and a stemmed corpus by applying the respective morphological transformation to the text documents that represent the video tutorials. We also singularized and stemmed the tags provided by the developers and built singularized and stemmed pyramids. We computed the pyramid scores using the corresponding pyramids and documents.

The average pyramid scores when recommending five and ten tags after using each morphological transformation are shown in columns 3–6 of Table 1. Our results are within the range of pyramid scores of three-tier pyramids for human annotations in general natural language tasks [21], which between 0.52 and 0.83. Our results are also higher than the average scores of a six-tier pyramid obtained by Haiduc et al. [14] when using LSI for term-based summaries of source code methods, which was around 0.3.

As seen in Table 1, we achieve the best results when using *singularization and BM25F(4,2,1)*. By using this combination, we can recommend tags that describe 66% and 62% of the information relevant to describe what the video is about when recommending five tags and ten tags, respectively. We also analyzed the effect of changing the number of tags recommended on the pyramid score of the tags when using singularization and stemming. We found that all the approaches exhibit the same behavior reported in Section 4.1 when recommending singularized and stemmed tags. Specifically, the score of the recommendations decreases as the number of tags recommended increases. Also, the most informative tags are recommended within the first five recommendations, and the pyramid score does not change drastically after the fifth tag is recommended.

4.3 RQ2 - Tag Provenance

Tagging is most useful when it provides information that is not readily available to the user. If the extracted tags are included in the title, they may not be useful as complementary information because the title is usually provided in the results of an online search. To establish whether our approach can be useful at providing novel information that is not readily available to the user, we analyze the provenance of all the relevant tags recommended by our best configuration, namely BM25F(4,2,1).

In total, 52.94% of all the tags recommended by BM25F(4,2,1) were relevant tags, i.e., tags that appear in any tier of a video's pyramid. For each video, we compiled the relevant tags extracted by BM25(4,2,1) and identified in which metadata field(s) each tag appeared[12]. We used this information to compute the percentage of relevant tags extracted from each metadata field, shown in Table 2. In particular, each cell represents the percentage of tags that appeared in the intersection of the fields given by the respective row and column. Note that the numbers in the diagonal represent the percentage of tags found exclusively each field. On average across all videos, 53.18% of the relevant tags appeared in the title of the video, 79.11% of them appeared in the description, and 95.53% of the tags appeared in the transcript. Additionally, none of the relevant tags appeared only in the title of the video, 1.43% of the tags appeared only in the description, while 15.43% of the tags appeared only in the audio transcript. Finally, 44.68% of the tags appeared in the intersection of all the fields of the video.

Table 2: Provenance of relevant tags extracted by BM25F(4,2,1) with singularization

	Title	Description	Transcript
Title	0.00%	3.04%	5.46%
Description	3.04%	1.43%	29.96%
Transcript	5.46%	29.96%	15.43%

Given that almost all of the relevant tags recommended by BM25F(4,2,1) appear in the transcripts and some appear only there, transcripts seem to be a crucial source of information when aiming for informative video tags. Moreover, the results indicate that extractive tagging approaches are a viable solution also in cases where the title and description are missing or not descriptive. Also, the high degree of tags that appear in both transcripts and description indicates that the content of the video is important not only for extracting relevant terms but also for increasing the relevance of informative terms that appear in the title and description.

Conversely, we computed the provenance of the tags in the ground truth – provided by the users in the study – in order to compare them with the provenance of the tags recommended by our best approach. These results, presented in allow us to explore if the provenance of the tags affects the performance of the tagging approaches for tagging software development videos.

On average across all videos, 26.05% of the tags provided by developers appeared in the title of the video, 49.17% of them appeared in the description, and 82.86% of the tags appeared in the transcript. An interesting finding is that 33.70% of the tags provided by the developers appear only in the transcript, whereas only 1.73% of

[12] After applying singularization

Table 3: Provenance of tags provided by developers

	Title	Description	Transcript
Title	0.29%	1.73%	2.93%
Description	1.73%	1.21%	25.13%
Transcript	2.93%	25.13%	33.70%

the tags appear only in the title or description. Moreover, 25.13% of the tags provided by developers appear in the combination of description and transcripts, and 21.10% of the tags appeared in the intersection of all the fields of the video, whereas 13.92% of the tags provided by developers do not appear in any of the metadata fields or the transcript of the video (i.e., abstractive terms).

By contrasting the provenance of the tags provided by developers with the tags produced by the best automatic approach, we see that developers tend to extract more terms that appear exclusively in the transcript. Also, they extracted more terms that appear in all the fields of a video compared to the best automatic approach.

Abstractive terms

Extractive tags are terms that are automatically extracted from the metadata fields of the video (i.e., title and description) or from its audio transcripts, whereas abstractive tags are terms that are not present in any of these sources [20].

Our results on the provenance of tags provided by developers showed that 13.92% of the tags provided by the participants in the study are abstractive and 86.08% of them are extractive. Our best performing approach, BM25F(4,2,1), is an extractive tagging approach that works by selecting tags from the available metadata (title and description) and the transcripts. Therefore, it is possible that the presence of the abstractive tags in the pyramids could lower the results of our best approach. In order to assess whether this is the case, we proceeded to build a version of the pyramids using only the tags given by the participants that are present in at least one of the metadata fields or the transcript. We then recomputed the pyramid scores for the different configurations of BM25F using these exclusively extractive pyramids.

Our results show that the average pyramid scores for all approaches are not heavily impacted when considering only the extractive tags. The average pyramid scores of the approaches increase slightly when we consider only extractive terms. In particular, the average pyramid score of BM25F(4,2,1) in combination with singularization increases from 0.6614 to 0.6625, when recommending five terms. These results are encouraging as they indicate that our approach is not negatively impacted by the presence of abstractive tags. Moreover, it shows that we can automatically extract tags that encompass up to 66% of the information that developers consider to be relevant to outline the content of a software development video using extractive tagging approaches.

We inspected the pyramids with and without abstractive tags and found that the abstractive tags are located in the lowest tier of the pyramid, in most cases only one developer used a particular abstractive tag. The low ranking of abstractive tags explains why the removal of abstractive tags from the ground truth rarely affects the optimal score of the pyramids.

In conclusion, the approach that performs the best in our experiments is the semi-structured Information Retrieval approach BM25F(4,2,1). The results indicated that the content of the video in the form of transcripts is a major contributor to the informativeness of the tags. In addition, the higher weights given to the title and description with respect to the transcript in the best performing approach are effective because both title and description are meant to provide an overview of the video. Thus, assigning more weight to these fields ensures that important terms from the metadata fields will be extracted. However, metadata fields are not enough to provide informative tags when the title and description are missing, uninformative, or used for SEO (e.g., tag-stuffing). In these cases, the terms of the transcript will aid in boosting relevant terms and decrease the weight of uninformative terms from the metadata fields.

As seen in our results, the best performing approach BM25F(4,2,1) can recommend the most informative tags within the first five tags. Moreover, the peak pyramid score of most of the extractive approaches in this paper is achieved when recommending the first tag. The first and second tags could prove useful for describing groups of related videos. In particular, when clustering similar videos, the first tag of each video in a cluster could be used to represent to a degree the content of the group of videos.

5 THREATS TO VALIDITY

Our study is not immune to threats to validity. In this section we describe the threats we faced and how we addressed them.

The threats to internal validity in our study include the impact of learning and fatigue effects on the quality of the participants' responses. To mitigate the fatigue effect, we only considered videos with a length of up to 7 minutes and distributed the videos such that the participants could complete the study within 25 to 45 minutes. Moreover, we recorded the time the participants spent annotating each video to ensure that the they spent a reasonable amount of time per video – enough time to watch the video entirely and write tags. Although we did not measure the participant's cognitive fatigue, research has shown that significant error rates appear after engaging in a task for longer than 60 minutes [24]. In order to mitigate the potential learning effect, we made sure the videos watched by each participant varied in topic and author.

As for threats to external validity, due to the size of the dataset, our results may not be generalizable to all the software development videos available. Nonetheless, we aimed at increasing the generalization of the findings for Java videos by including videos on a wide variety of topics and created by multiple creators. Moreover, our approach applies to any existing video, yielding a reasonable performance also when no transcripts are available.

Threats to construct validity refer to how we measured the effectiveness of the approaches. We mitigated threats to construct validity by employing the pyramid method, which is a well establish measurement approach that has been successfully used in the fields of Information Retrieval [21] and Software Engineering [14, 29]. Moreover, the pyramid score provides a reliable assessment of content selection quality in tagging and summarization where there are multiple annotations available, as in our study.

6 RELATED WORK

Our work is closely related to several fields. Therefore, we divide the related work into three categories. First, we outline the work on keyword extraction and term-based summarization in Software

Engineering. Second, we present work on the analysis of software development videos.

6.1 Keyword Extraction and Term-based Summarization in Software Engineering

Keyword extraction and term-based summarization techniques have been widely used in Software Engineering research to describe, summarize, categorize, and increase the discoverability of a wide variety of software artifacts (e.g., source code [9, 11, 14], software repositories [35], Q&A posts [30], developer communications [23], and change requests [10]). Moreover, these techniques have been applied to support developers in a wide variety of tasks, such as software summarization [9, 11, 14], software documentation [23, 33], and bug triage [43]. However, there is no work on applying term-based summarization techniques to obtain keywords that describe the content of software development videos.

Research efforts have aimed a tagging Q&A posts on StackOverflow [1, 4, 16, 30, 34, 37, 39, 40, 44]. Stanley and Byrne [34] developed a model to predict the tags that users assign to a post. Several tag propagation approaches that leverage tags in existing Q&A posts to produce tags for new posts have been proposed, Xia et al. [39, 40] present TagCombine, a multi-label ranking component which considers tag recommendation as a multi-label learning problem. TagCombine uses the relationship between different terms and tags, to recommend tags. Similarly, Wang et al. [37] developed EnTagRec. EnTagRec uses a statistical model based on historical tag assignments to recommend tags for new Q&A posts. More recently, Zhou et al. [44] introduced TagMulRec, an approach that assigns a relevance score to tags based on the similarity of the posts currently labeled with it and the new post. We used TagMulRec as one of the approaches on the paper as it outperforms all the other existing approaches on SE Q&A tagging.

Tagging software development videos is different from tagging StackOverflow posts in two ways: the lack of predefined tags and the existence of additional information that is not available on StackOverflow posts.

In addition to tagging, keyword extraction approaches have been applied to StackOverflow posts to obtain additional information from the posts [5, 26, 31, 36]. Barua et al. [5] used LDA to capture the main topics discussed in Stack Overflow posts automatically. Similarly, Linares-Vasquez et al. [17] used LDA to extract the topics of posts discussing Android development. Rigby and Robillard [31] present a classifier that uses IR to obtain terms that describe source code elements from Q&A posts.

6.2 Analysis of Software Development Videos

Software development videos have been recently studied as a source of information to support software developers [19, 27]. MacLeod et al. [19] report the results of a set of interviews with creators of software development videos. The creators stated that they "make videos to document what they wished they had known before they started a task" and to "spare others from having to go through the same discovery process." Ponzanelli et al. [28] present CodeTube, a search engine for software development videos that enables developers to search and view specific fragments of software development videos, whereas, Yadid and Yahav [41] present ACE, a tool that combines language models and image processing techniques to extract source code from software development videos. More recently, Poché [25] leveraged the term frequency of words in the comments of software development videos to identify and extract general user concerns about the video.

7 CONCLUSIONS

In this paper, we present an evaluation of tagging approaches towards recommending relevant terms to label software development videos. We evaluate a total of 19 configurations across seven different tagging approaches in a user study conducted with 57 developers and 70 software development videos. The results of our study show that BM25F, a semi-structured IR approach, can produce tags that can describe up to 66% of the information that developers perceive to be important about software development videos.

We found that the informativeness of tags obtained by automatic approaches decreases as the number of tags increases. However, it does not significantly decay after the first five recommendations. We believe five tags to be a good start for automatic tagging purposes, despite the fact that participants suggested an average of ten tags per video. This is due to the fact that five tags led to better results than ten in our evaluation and five tags have also been proven to be useful in the case of StackOverflow.

The place of origin of tags used by developers and recommended by the best tagging approach indicate that the underlying information in the transcripts of the video is vital and needs to be exploited when creating tags. We also found that developers use a combination of extractive and abstractive tags to describe software development videos, with a broader use of extractive tags. Moreover, the abstractive tags they used tend to be different from developer to developer. Therefore, they are unlikely to have a large impact on the evaluation of extractive summarization approaches. This finding suggests that work is needed on the improvement of extractive tags as well as on the generation of abstractive tags.

We believe that our results on the generation of automatic tags can potentially reduce the manual effort involved in searching and discovering relevant videos for a developers' information need. The presence of informative tags can enable developers to make a quick assessment of the content of the video and their relevance to their information need without having to watch the entire video or read a long description.

Our future work will focus on improving the IR techniques we found to work best and complementing them with software-specific information. In particular, we plan on using glossaries to identify and explicitly include software-specific aspects in the tags, such as APIs found in the video, common errors, the IDEs used, etc. Also, we will further explore abstractive summarization techniques to complement the information provided in tags. Additionally, while current automatic transcripts lack in quality, we plan to improve automatic speech-to-text transcriptions for software development videos.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation grant CCF-1526929. The authors thank Annibale Panichella for providing the implementation of LDA-GA.

REFERENCES

[1] Miltiadis Allamanis and Charles Sutton. 2013. Why, When, and What: Analyzing Stack Overflow Questions by Topic, Type, and Code. In *Proceedings of the 10th IEEE Working Conference on Mining Software Repositories (MSR'13)*. IEEE, San Francisco, CA, USA, 53–56.

[2] Hrishikesh Aradhye, George Toderici, and Jay Yagnik. 2009. Video2Text: Learning to Annotate Video Content. In *Proceedings of the 9th IEEE International Conference on Data Mining Workshops (ICDMW'09)*. IEEE, Miami, FL, USA, 144–151.

[3] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley.

[4] Ebrahim Bagheri and Faezeh Ensan. 2016. Semantic Tagging and Linking of Software Engineering Social Content. *Automated Software Engineering* 23, 2 (March 2016), 147–190.

[5] Anton Barua, Stephen Thomas, and Ahmed Hassan. 2012. What Are Developers Talking About? An Analysis of Topics and Trends in Stack Overflow. *Empirical Software Engineering* 19, 3 (Nov. 2012), 619–654.

[6] David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet Allocation. *The Journal of Machine Learning Research* 3 (Jan. 2003), 993–1022.

[7] Stefan Buttcher, Charles Clarke, and Gordon Cormack. 2010. *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press.

[8] Leonardo Canseco, Lori Lamel, and Jean-Luc. Gauvain. 2005. A Comparative Study Using Manual and Automatic Transcriptions for Diarization. In *Proceedings of the 4th IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'05)*. IEEE, Cancún, Mexico, 415–419.

[9] Andrea De Lucia, Massimiliano Di Penta, Rocco Oliveto, Annibale Panichella, and Sebastiano Panichella. 2012. Using IR Methods for Labeling Source Code Artifacts: Is It Worthwhile?. In *Proceedings of the 20th IEEE International Conference on Program Comprehension (ICPC'12)*. IEEE, Passau, Germany, 193–202.

[10] Giuseppe A. Di Lucca, Massimiliano Di Penta, and Sara Gradara. 2002. An Approach to Classify Software Maintenance Requests. In *Proceedings of the 18th IEEE International Conference on Software Maintenance (ICSM'02)*. IEEE, Montreal, Canada, 93–102.

[11] Brian Eddy, Jeffrey Robinson, Nicholas Kraft, and Jeffrey Carver. 2013. Evaluating Source Code Summarization Techniques: Replication and Expansion. In *Proceedings of the 21st IEEE International Conference on Program Comprehension (ICPC'13)*. IEEE, San Francisco, CA, USA, 13–22.

[12] Javier Escobar-Avila, Esteban Parra, and Sonia Haiduc. 2017. Text Retrieval-based Tagging of Software Engineering Video Tutorials. In *Proceedings of the 39th IEEE/ACM International Conference on Software Engineering (ICSE'17)*. IEEE, Buenos Aires, Argentina, 341–343.

[13] Laura Granka, Thorsten Joachims, and Geri Gay. 2004. Eye-tracking Analysis of User Behavior in WWW Search. In *Proceedings of the 27th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR'04)*. ACM, Sheffield, UK, 478–479.

[14] Sonia Haiduc, Jairo Aponte, Laura Moreno, and Andrian Marcus. 2010. On the Use of Automated Text Summarization Techniques for Summarizing Source Code. In *Proceedings of the 17th IEEE Working Conference on Reverse Engineering (WCRE'10)*. IEEE, Beverly, MA, USA, 35–44.

[15] Karen Sparck Jones. 1972. A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation* 28, 1 (Jan. 1972), 11–21.

[16] Arash Joorabchi, Michael English, and Abdulhussain Mahdi. 2015. Automatic Mapping of User Tags to Wikipedia Concepts: The Case of a Q&A Website Ű StackOverflow. *Journal of Information Science* 41, 5 (May 2015), 570–583.

[17] Mario Linares-Vasquez, Bogdan Dit, and Denys Poshyvanyk. 2013. An Exploratory Analysis of Mobile Development Issues Using Stack Overflow. In *Proceedings of the 10th IEEE Working Conference on Mining Software Repositories (MSR'13)*. IEEE, San Francisco, CA, USA, 93–96.

[18] Craig Macdonald, Richard McCreadie, Rodrygo LT Santos, and Iadh Ounis. 2012. From Puppy to Maturity: Experiences in Developing Terrier. *Open Source Information Retrieval* (Aug. 2012), 60–63.

[19] Laura MacLeod, Margaret-Anne Storey, and Andreas Bergen. 2015. Code, Camera, Action: How Software Developers Document and Share Program Knowledge Using YouTube. In *Proceedings of the 23rd IEEE International Conference on Program Comprehension (ICPC'15)*. 104–114.

[20] Mark Melenhorst, Marjan Grootveld, Mark van Setten, and Mettina Veenstra. 2008. Tag-based Information Retrieval of Video Content. In *Proceedings of the 1st International Conference on Designing Interactive User Experiences for TV and Video (UXTV'08)*. ACM, Silicon Valley, CA, USA, 31–40.

[21] Ani Nenkova and Rebecca Passonneau. 2004. Evaluating Content Selection in Summarization: The Pyramid Method. In *Proceedings of the 3rd Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL'04)*, Vol. 4. ACM, Boston, MA, USA, 145–152.

[22] Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshynanyk, and Andrea De Lucia. 2013. How to Effectively Use Topic Models for Software Engineering Tasks? An Approach Based on Genetic Algorithms. In *Proceedings of the 35th ACM/IEEE International Conference on Software Engineering (ICSE'13)*. IEEE, San Francisco, CA, USA, 522–531.

[23] Sebastiano Panichella, Jairo Aponte, Massimiliano Di Penta, Andrian Marcus, and Gerardo Canfora. 2012. Mining Source Code Descriptions From Developer Communications. In *Proceedings of the 20th IEEE International Conference on Program Comprehension (ICPC'12)*. IEEE, Passau, Germany, 63–72.

[24] Nathalie Pattyn, Xavier Neyt, David Henderickx, and Eric Soetens. 2008. Psychophysiological Investigation of Vigilance Decrement: Boredom or Cognitive Fatigue? *Physiology & Behavior* 93, 1 (Jan. 2008), 369–378.

[25] Elizabeth Heidi Poché. 2017. *Analyzing User Comments On YouTube Coding Tutorial Videos*. mathesis. Louisiana State University, Baton Rouge, LA, USA.

[26] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. 2013. Seahawk: Stack Overflow in the IDE. In *Proceedings of the 35th ACM/IEEE International Conference on Software Engineering (ICSE'13)*. IEEE, San Francisco, CA, USA, 1295–1298.

[27] Luca Ponzanelli, Gabriele Bavota, Andrea Mocci, Massimiliano Di Penta, Rocco Oliveto, Mir Hasan, Barbara Russo, Sonia Haiduc, and Michele Lanza. 2016. Too Long; Didn't Watch!: Extracting Relevant Fragments from Software Development Video Tutorials. In *Proceedings of the 38th ACM/IEEE International Conference on Software Engineering (ICSE'16)*. Austin, TX, USA, 261–272.

[28] Luca Ponzanelli, Gabriele Bavota, Andrea Mocci, Massimiliano Di Penta, Rocco Oliveto, Barbara Russo, Sonia Haiduc, and Michele Lanza. 2016. CodeTube: Extracting Relevant Fragments from Software Development Video Tutorials. In *Proceedings of the 38th ACM/IEEE International Conference on Software Engineering (ICSE'16)*. ACM, Austin, TX, USA, 645–648.

[29] Sarah Rastkar, Gail C. Murphy, and Gabriel Murray. 2010. Summarizing Software Artifacts: A Case Study of Bug Reports. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE'10)*. ACM, Cape Town, South Africa, 505–514.

[30] Smrithi Rekha, N. Divya, and Sivakumar Bagavathi. 2014. A Hybrid Auto-tagging System for StackOverflow Forum Questions. In *Proceedings of the 1st International Conference on Interdisciplinary Advances in Applied Computing (ICONIAAC'14)*. ACM, Amritapuri, India, 1–5.

[31] Peter Rigby and Martin Robillard. 2013. Discovering Essential Code Elements in Informal Documentation. In *Proceedings of the 35th ACM/IEEE International Conference on Software Engineering (ICSE'13)*. IEEE, San Francisco, CA, USA, 832–841.

[32] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trendső in Information Retrieval* 3, 4 (April 2009), 333–389.

[33] Paige Rodeghero. 2016. Discovering Important Source Code Terms. In *Proceedings of the 38th ACM/IEEE International Conference on Software Engineering (ICSE'16)*. ACM, Austin, TX, USA, 671–673.

[34] Clayton Stanley and Michael Byrne. 2013. Predicting Tags for StackOverflow Posts. In *Proceedings of 12th International Conference on Cognitive Modelling (ICCM'13)*. Carleton, CA, 414–419.

[35] Kai Tian, Meghan Revelle, and Denys Poshyvanyk. 2009. Using Latent Dirichlet Allocation for Automatic Categorization of Software. In *Proceedings of the 6th IEEE Working Conference on Mining Software Repositories (MSR'09)*. IEEE, Vancouver, Canada, 163–166.

[36] Yuan Tian, David Lo, and Julia Lawall. 2014. SEWordSim: Software-specific Word Similarity Database. In *Companion Proceedings of the 36th IEEE/ACM International Conference on Software Engineering (ICSE'14)*. ACM, Hyderabad, India, 568–571.

[37] Shaowei Wang, David Lo, Bogdan Vasilescu, and Alexander Serebrenik. 2014. EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites. In *Procedings of the 30th IEEE International Conference on Software Maintenance and Evolution (ICSME'14)*. IEEE, Victoria, BC, Canada, 291–300.

[38] Francisco De Sousa Webber. 2015. Semantic Folding Theory And its Application in Semantic Fingerprinting. *CoRR* abs/1511.08855 (2015).

[39] Xin Xia, David Lo, Xinju Wang, and Bo Zhou. 2013. Tag Recommendation in Software Information Sites. In *Proceedings of the 10th IEEE Working Conference on Mining Software Repositories (MSR'13)*. IEEE, San Francisco, CA, USA, 287–296.

[40] Xin Xia, David Lo, Xin-Yu Wang, and Bo Zhou. 2015. TagCombine: Recommending Tags to Contents in Software Information Sites. *Journal of Computer Science and Technology* 30, 5 (Sept. 2015), 1017–1035.

[41] Shir Yadid and Eran Yahav. 2016. Extracting Code from Programming Tutorial Videos. In *Proceedings of the 6th ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!'16)*. ACM, Amsterdam, The Netherlands, 98–111.

[42] YouTube. [n. d.]. Statistics - YouTube. ([n. d.]). https://www.youtube.com/yt/press/statistics.html

[43] Tao Zhang, He Jiang, Xiapu Luo, and Alvin Chan. 2015. A Literature Review of Research in Bug Resolution: Tasks, Challenges and Future Directions. *Comput. J.* (Dec. 2015), 741–773.

[44] Pingyi Zhou, Jin Liu, Zijiang Yang, and Guangyou Zhou. 2017. Scalable Tag Recommendation for Software Information Sites. In *Proceedings of the 24th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER'17)*. IEEE, Campobasso, Italy, 272–282.