Titre: Title:	TechTube: Summarizing Relevant Parts from Technical Videos
Auteur: Author:	Mahmood Vahedi
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Vahedi, M. (2021). TechTube: Summarizing Relevant Parts from Technical Videos [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/6290/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6290/
Directeurs de recherche: Advisors:	Foutse Khomh, & Giuliano Antoniol
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

TechTube: Summarizing Relevant Parts from Technical Videos

MAHMOOD VAHEDI
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Mai 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

TechTube: Summarizing Relevant Parts from Technical Videos

présenté par **Mahmood VAHEDI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Daniel ALOISE, président
Foutse KHOMH, membre et directeur de recherche
Giuliano ANTONIOL, membre et codirecteur de recherche
Mohammad HAMDAQA, membre

ACKNOWLEDGEMENTS

Most importantly, I am delighted to have had Prof. Foutse Khomh of the Computer and Software Engineering department as my thesis advisor and Prof. Giuliano Antoniol of the Computer Software Engineering department as my co-supervisor. I would like to extend my gratitude to members of the jury, who reviewed my thesis. Many thanks to my heroes, Mom and Dad who forwent many tribulations to let me have an education. Also, I would like to thank my siblings, in-laws and the best friends who have supported me in different manners through this joyful journey.

RÉSUMÉ

Les développeurs de logiciels sont confrontés à plusieurs problèmes techniques indissociables de la programmation. L'approche la plus importante qu'ils utilisent pour faire face à de tels problèmes consiste à regarder des tutoriels en ligne.

Les vidéos de didacticiel sont très diverses et toutes sortes d'entre elles peuvent être trouvées sur Internet. Par exemple, YouTube contient désormais des millions de vidéos de bonne qualité sur des sujets techniques, tels que la programmation. Cependant, malgré la richesse de leur contenu, ces vidéos sont parfois trop longues et tous les sujets couverts ne sont pas toujours pertinents pour les développeurs cherchant des réponses à des questions précises. Ainsi, identifier et résumer les fragments pertinents de ces vidéos pourrait faire gagner du temps et des efforts précieux aux développeurs. Dans ce mémoire, nous proposons une nouvelle technique appelée TechTube. Cette technique peut être utilisée pour trouver des segments vidéo pertinents pour une tâche technique donnée.

TechTube permet aux développeurs de formuler l'objet de leur recherche sous forme de requêtes en langage naturel. Ces requêtes sont ensuite enrichies grâce aux techniques de reformulation de requêtes, développées dans le domaine de la recherche d' information. La sortie de TechTube est une séquence de segments vidéo pertinents pour la tâche à accomplir. Contrairement aux travaux de recherche précédents, notre approche divise la vidéo en détectant le silence dans les pistes audio de la vidéo.

Des expériences utilisant 98 requêtes de recherche liées à la programmation montrent que notre approche fournit les vidéos pertinentes parmi les 5 premiers résultats 93 % du temps avec une précision moyenne de 76 %. Nous constatons également que TechTube peut fournir la section la plus pertinente d'une vidéo technique avec une précision de 67 % et un rappel de 53 %, ce qui surpasse les approches de l'état de l'art. Nous avons réalisé une étude d'utilisabilité avec 16 participants, afin d'évaluer la pertinence des résumés vidéo générés. Ces participants ont rapporté que les résumés vidéo générés par TechTube sont précis, concis, et très utiles pour leurs tâches de programmation. Ils ont aussi jugé ces vidéos plus utiles que les vidéos complètes originales.

ABSTRACT

During development and maintenance activities, developers often turn to technical online videos (i.e., tutorials) to learn new concepts and find solutions to some technical problems.

Tutorial videos are extensively diverse and the number of online videos is growing at a very fast pace. For example, YouTube now contains millions of high-quality videos on technical topics, such as programming. However, despite the convenience and in some cases popularity, the audiovisual explanations of the videos might also claim more time from the developers than the text-only materials (*e.g.,* programming Q&A threads). In fact, unlike the programming Q&A threads, programming videos might not be fully indexed or well organized e.g., structured as explicit questions and answers. They might also contain noisy, redundant information that might be of little interest to the developers. Thus, the developers often attempt to find out the relevant sections by skimming (i.e., forward and backward the video stream) through the videos. However, skimming through a video is often more time-consuming than scrolling through a regular text-based website. Thus, pinpointing the relevant sections within a technical video and guiding developers to these sections has the potential to save their valuable time and effort.

In this thesis, we propose a novel technique called TechTube. This technique can be used to find video segments that are relevant to a given technical task.

TechTube allows a developer to express the task as a natural language query. As it might happen to every software developer, some words would be missed in the queries. To account for missing words in the query, TechTube automatically reformulates the query using techniques based on information retrieval. The reformulated query is matched against a repository of online technical videos. The output from TechTube is a sequence of relevant video segments that can be useful to implement the task at hand. Unlike previous researches, our approach splits the video by detecting silence in video audio tracks.

Experiments conducted using 98 programming related search queries show that our approach delivers the relevant videos within the Top-5 results 93% of the time with a mean average precision of 76%. We also find that TechTube can deliver the most relevant section of a technical video with 67% precision and 53% recall, outperforming the state of the art approaches. We conducted a user study with 16 participants to assess the effectiveness of summaries generated by TechTube. The participants reported that video summaries generated by TechTube are accurate, precise, concise, and very useful for their programming tasks. They also found the summaries to be more useful than the original complete videos.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

Q&A	Questions and Answers
SO	Stack Overflow
API	Application Programming Interface
OCR	Optical Character Recognition
GT	Ground Truth
RQ	Research Question
IT	Information Technology
TF	Term Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
PR	Page Rank
BM25	Best Matching 25
FFMPEG	Fast Forward Moving Pictures Expert Group
SE	Software Engineering
LCS	Longest Common Subsequence
MRR	Mean Reciprocal Rank
MAP	Mean Average Precision
RR	Reciprocal Rank
AP	Average Precision
TT	TechTube
BL	Base Line
VM	Virtual Machine
IDE	Integrated Development Environment
GUI	Graphical User Interface
HMM	Hidden Markov Model
SNR	Signal to Noise Ratio
SVM	Support Vector Machine
KNN	K Nearest Neighbors
NB	Naive Bayes
MIR	Music Information Retrieval
AAS	Automatic Audio Segmentation
MOF	Means Of Frames
AC	Agglomerative Clustering
DTW	Dynamic Time Wrapping

FIR	Finite Impulse Response
GMM	Gaussian Mixture Model
MFCC	Mel Frequency Cepstral Coefficients
ZCR	Zero Crossing Rate
LPC	Linear Prediction Coefficients
LSP	Linear Spectral Pairs
PRF	Pseudo Relevance Feedback
LSI	Latent Semantic Indexing
SITIR	SIngle Trace and Information Retrieval
QE	Query Expansion
HAL	Hyperspace Analogue to Language
LSA	Latent Semantic Analysis
GLSA	Generalized Latent Semantic Analysis
ESA	Explicit Semantic Analysis
CLESA	Cross-Language Explicit Semantic Analysis
PMI-IR	Pointwise Mutual Information - Information Retrieval
SCO-PMI	Second-order CO-occurence - Pointwise Mutual Information
NGD	Normalized Google Distance
DISCO	Extracting DIStributionally similar words using CO-occurences
CBOW	Continous Bag Of Words
MLE	Maximum-Likelihood Estimation
PQV	Pseudo Query Vector
NLTM	Neural Language Translation Model
GLM	Generalized Language Model
DFR	Divergence From Randomness
MI	Mutual Information
RM	Relevance Model

CHAPTER 1 INTRODUCTION

Studies suggest that software developers spend about 20% of their development time in searching for programming solutions on the Internet [1]. In some cases, the problems are quite challenging. Besides programming Q&A websites (*e.g.,* Stack Overflow) and software repositories (*e.g.,* GitHub, SourceForge), they frequently look for relevant online technical videos which are sometimes very difficult and in some cases disappointing for the developers.

YouTube is one of the most popular video websites on the Internet, many users (not only software developers) search for videos on this platform. YouTube offers millions of high-quality technical videos, including programming tutorials [2].

Comparing videos tutorial to text-only resources, one observes that video tutorials capture the finer details of a programming task through audiovisual information, annotations, and also non-verbal cues. Therefore programming video tutorials can often represent a better and faster choice for developers than the traditional text-only resources (*e.g.,* API documentation) [3].

However, searching through video tutorials can be challenging. Unlike the programming Q&A threads, programming videos might not be fully indexed or well organized *e.g.,* structured as explicit questions and answers. They might also contain noisy, redundant information that might be of little interest to the developers [4]. Developers using video tutorials often attempt to find out the relevant sections by skimming (*i.e.,* forward and backward the video stream) through the videos. However, skimming through a video is often more time-consuming than scrolling through a regular text-based website. Hence, pinpointing the relevant sections within a technical video and guiding developers to these sections has the potential to save them valuable time and effort.

There have been a few existing studies [4, 5, 6] that attempted to extract important sections from online technical videos. Adcock et al. [5] capture keywords from the slides of a webcast using Optical Character Recognition (OCR) and then help to locate the important slides using text processing and Information Retrieval methods (*e.g.,* Lucene). This approach heavily relies on OCR technology to collect the textual contents, which results in quite a poor performance with low-quality or noisy videos. Ponzanelli et al. [4] suggest relevant sections from YouTube videos where they use textual contents from the video frames containing code segments to split their videos. Therefore, it is clearly possible that their approach could fail to split the technical videos that do not contain any code segments (*e.g.,* tool installations, webinars). Furthermore, software development is not exclusive to any specific programming

languages which means their algorithms and heuristics that are designed to detect the Java code might not be suitable for other programming languages (*e.g.,* Python).

1.1 Research Objectives

In this thesis, we propose a novel approach – TechTube – that identifies the most relevant sections from a technical video and delivers coherent, concise, and relevant video summaries against a natural language query. We assess and contrast the performance of TechTube with the performance of existing approaches from the literature. We also assess the relevance of summaries generated by TechTube through a user study.

TechTube allows a developer to articulate a query, representing the task at hand, in natural language. To account for missing terms and improve recall, our approach automatically augments the query by reformulating it using popular pseudo relevance feedback techniques (*e.g.,* Rocchio [7]). The reformulated query is then matched against a repository of online technical videos to retrieve the top-5 most related videos. The output from TechTube is a sequence of *relevant* video segments (a.k.a., video summary) that come in handy for implementing the task at hand, learning the topic being searched, and address complex problems.

We assess the effectiveness of TechTube by answering the following research questions:

- RQ_1: How does TechTube perform in (a) retrieving the relevant technical videos against natural language queries, and (b) identifying the most relevant sections from these videos?

TechTube's video repository can be populated with videos from any available online site. Hence, we need a search engine that retrieves the most appropriate videos and the most related summary of the videos. In order to provide TechTube with the best setup, we experiment with two different search engines: (a) Ad-hoc search engine that works based on measuring the text similarity using the Cosine similarity method. (b) Apache Lucene which is measuring the text similarity using the BM25 method. We answer the following research question.

- RQ_2: How does Apache Lucene perform in retrieving the relevant videos? Is its use justified?

When summarizing videos, TechTube performs a segmenting process on the audio file of the videos. To ensure good efficiency, we experiment with two segmentation techniques. The first

technique uses the available auto-generated subtitle of the video downloaded from YouTube and segment it based on YouTube's timing. The other one extracts the available speech using speech recognition techniques and segments them based on the available silences (*i.e.,* speaker pause in the speech). We compare the performance of these two segmentation techniques by answering the following research question.

- RQ_3: Is the use of silence-based splitting of videos justified?

One of the main components of TechTube is its query reformulation module. We assess the effectiveness of our query reformulation mechanism by answering the following research question.

- RQ_4: Does the query reformulation mechanism improve (a) the retrieval of relevant video and–or (b) the retrieval of relevant fragment of a video?

To assess the benefits of TechTube for professional developers, we conduct a user study to examine how developers perceive the accuracy, preciseness, conciseness, and usefulness of the generated summaries. We answer the following research question.

- RQ_5: Does TechTube deliver relevant, concise, useful video contents against the developers' queries?

In summary, this thesis makes the following contributions:

(a) A novel approach to identify the most relevant section of a technical video to a given natural language query.

(b) Comprehensive evaluation of the proposed approach and a comparison with state of the art approaches.

(c) A replication package [8] of our approach – TechTube – for the replication and third-party reuse.

1.2 Thesis Plan

The rest of this thesis is organized as follows. In Chapter 2 we review the related literature. In Chapter 3 we give a general overview of concepts that have been used in this study. In chapter 4, we describe the framework implemented through this study. In Chapter 5 we discuss the results of our case study that aimed at evaluating the performance of our proposed solution and justifying the modules used through development. In chapter 6 we present a summary of our work and outlines some avenues for future works.

CHAPTER 2 LITERATURE REVIEW

Video Processing

The work of MacLeod et al. [3] is one of the first few that investigated the reason for producing video tutorials by the users. Firstly, they scrutinized the manner of sharing technical development knowledge by means of analyzing an extensive set of YouTube video tutorials and screencasts pertinent to the programming task. Afterward, they provided the pros and cons of the technical screencasts by interviewing the video tutorial generators. By concentrating on the screencasts, they discerned that the video tutorials can be complementary resources for the text-based resources (*e.g.,* API documentation). Since the video tutorials benefit from the audiovisual trait, they are valuable materials for sharing the knowledge between two developers. MacLeod et al. [3] specified this significant feature as the key advantage of the video tutorials over other written documents. Their study motivates us to perform research on the general set of video tutorials containing different task implementation topics *e.g.,* application development, software installation, API explanation, and etc.

Escobar-Avila et al. [9] conduct a survey to investigate the preference of their participants (*i.e.,* university students, practitioners) in learning topics related to programming and in a more general set, concerning computer science. Their survey provides evidence that human subjects prefer video tutorials (*i.e.,* resources with visual features) as their main resource for learning a new topic. To gain deeper insights into this preference, they extended their study to examine the main characteristics of the video tutorials preferred by the participants. Their results show that participants mostly liked using multiple small online videos for learning a new topic rather than a video with a long time duration. Furthermore, Escobar-Avila et al. [9] also asked the participants for suggestions that could help improve the quality of video tutorials. Their work motivates the work presented in this thesis, which aims to reduce long video tutorials into video summaries, by retaining only their essential sections.

Adcock et al. [5] proposed a tool–TalkMiner– that creates a search list of words in the slides provided in the source video. The principal objective of the tool is to extract the unique slide images based on the search query entered. There were few challenges that could reduce the reliability of a simple automatic algorithm such as, the poor quality of the videos, conference room darkness, scrolling and zooming the slides images, cameras' angle, picture-in-picture videos containing the main slides and a small image of the speaker on the corner. To address this issue, Adcock et al. [5] developed an automatic slide identification algorithm that, to

a greater extent, could overcome the aforementioned challenges. The proposed algorithm relies on OCR (*i.e.,* Optical Character Recognition), to capture keywords from the slides of a webcast, and provides the critical slides from the webcast by utilizing a combination of OCR, text processing, and information retrieval methods (*e.g.,* Lucene). However, because of its high dependency on OCR technology, this approach might not perform well on poor-quality and noisy videos. Their work motivated us to leverage another feature of the video (*e.g.,* speech track) in recognizing the essential sections.

A study by Yadid and Yahav [10] proposed a search engine that retrieves the most related programming videos by identifying the code snippets provided as the content, inside the video frames. Their approach retrieves relevant parts of videos by pointing out the exact part of the video containing the code of interest [10]. This proposed approach has two main limitations. Because of the gradual nature of code editing and lack of full visibility on the complete code (*i.e.,* code snippets are shown partly in different video frames), identifying a complete code fragment, to a certain extent, is difficult. Furthermore, due to the different characteristics of programming-specific videos such as the variety of font sizes, colors, line brakes in IDE (*i.e.,* Integrated Development Environment) or text editors, dealing with text recognition methods such as OCR (*i.e.,* Optical Character Recognition) is complex and inaccurate. Yadid and Yahav [10] also proposed an OCR-based approach that mixed statistical language models to extract the code snippets from image frames. Despite the relatively good performance of the approach (i.e., average precision between 80% and 82%), their work covers only videos related to programming containing code snippets. Moreover, it cannot be applied to other type of technical video tutorials (*e.g.,* tool installation, webinars). This limitation motivates us to provide a generalized framework covering different task implementation technical videos.

Ponzanelli et al. [4] concentrate on the programming video tutorials besides online Q&A threads as complementary resources for problem solving and learning new topics. They proposed to identify the relevant fragments of the video tutorials and divide the video tutorials into sections based on the video frames content with the corresponding audio transcript. Their approach also provides top related Q&A threads from Stack Overflow in addition to the online technical fragments of the videos. Their approach first, identifies the video frames containing code snippets. It then extracts all the available contents (*i.e.,* texts which are composed of words) making use of optical character recognition. Ponzanelli et al. [4] take advantage of dictionary-based filtering to ignore the invalid English keywords. Afterward, their approach identifies the Java code existing inside the video frames with the help of shape detection methods in order to identify the development environment, and H-AST (Heterogenous Abstract Syntax Tree). Finally, by converting the video frames to 8-bit grayscale images and using LCS (Longest Common Sub-string), CodeTube recognizes the Java code available

inside the video frames.

This approach employs OCR to extract the content of the video frames (*e.g.,* code snippets). Similar to us, they also apply the audio transcript beside the frame contents in order to classify the relevant fragments of the video. Given that their approach relies on detecting code snippets in the video tutorials, it can be limited to video tutorials which mostly contain code snippets. Furthermore, their algorithms and heuristics that are designed to detect the Java code might also not be suitable for other programming languages (*e.g.,* Python).

Hauptmann and Smith [11] built a digital video library for better indexing and retrieval of the videos. They processed image frames of the videos while they were doing the same procedure for the audio signal and word relevance. They first, extract the speech of the videos using Sphinx-II by applying a model called the semi-continuous Hidden Markov Model (HMM). afterward, it is used to analyze the audio track of the video. They perform this process using Signal to Noise Ratio (SNR) in order to identify the speech acoustic paragraph by making use of the difference that exists in the voice energy (*i.e.,* considering the low energy point as the silence). They identify break-points in a video by analyzing the image histogram and optical flow. They use these break-points to infer topics. They process the video frames to point out the acoustic paragraphs extracted from audio segmenting. For extracting the important keywords of the acoustic paragraphs they used TF-IDF and match them against the processed input natural language search query of the users and retrieve the most relevant section of the videos. Due to the use of keyword identification for acoustic paragraphs and matching with natural language search queries, their approach is not considering the segments that could be relevant to the queries conceptually but not literally. Accordingly, we got motivated to overcome such a challenge by taking advantage of the Longest Common Subsequence (LCS) technique.

Similarly to Hauptmann and Smith [11], Boreczky and Wilcox [12] have used HMM to classify audio chunks when segmenting videos. Specifically, they used computed image differences (*i.e.,* frames histogram analysis), audio difference (*i.e.,* low energy of the neighbors to identify the acoustic audio points), and object motion (*i.e.,* identifying the time and the place the images are zoomed or panned) to segment the videos.

Jincheng Huang et al. [13] proposed break detectors for videos' audio track, image colors, and frame motions. Their approach computes a set of 12 audio features for each second of videos to construct a dissimilarity index. They detect the motion of the video frames by comparing the phase correlation for each two neighbor video frames. On the one hand, they detect the color break-point of the frames by comparing the color histograms of neighbor frames. Finally, they extract the video segments by integrating their audio visual processes

[13]. This approach may not perform well for software engineering (SE) and computer science (CS) technical videos, because of its high dependency on image frame processing.

Chávez et al. [14] investigated the segmenting of videos using supervised learning. Their approach classifies each video frame into different classes *e.g.,* common shot frame, cut frame, fade frame, dissolve frame. Firstly, they calculate the color histogram difference of every two neighbor frames. Secondly, they compute the diversity of frames by their 8*8 block-wise histogram. Thereupon, they build a feature vector for each frame to have the current frame property in order to improve the performance of the shot frame detection. Then, Chávez et al. [14] distinguish the non-cut frames from the cut frames by performing a binary comparison of their corresponding feature vectors. Eventually, they classify shot frames and the gradual transition frames. [14] To select a suitable classification technique for their approach, they compared the performance of K-Nearest Neighbors (KNN), Naive Bayes (NB), and Support Vector Machine (SVM) and found that the K-Nearest Neighbors algorithm outperforms the other algorithms with better precision, recall, and F1-score.

Similar to the approach proposed by Jincheng Huang et al. [13], this approach may not work well on software engineering or computer science technical video tutorials. The huge variety of technical video tutorials in the mentioned fields are picturing the code implementation. Consequently, most of the time there is not much frame difference to distinguish each neighbor frame.

Query Reformulation

Bajracharya and Lopes [15] examined search queries on the Koders platform and compared them to the queries of developers searching on the web. They discovered that developers use 30,000 search queries per day and that their performed queries have two characteristics. Queries are very short and the terms in queries are quite diverse. On the Koders platform, Bajracharya and Lopes [15] discovered that 76% of all the input queries were modified by their users, which suggests that developers' search queries often have to be reformulated. Hence, when designing TechTube we included a query reformulation mechanism.

Furnas et al. [16] state that there are vocabulary problems (*i.e.,* using conceptually wrong words) inside natural language search queries of intermittent users which lead them to failure; missing the best possible outcome. Processing users' input query is challenging. This is because an abstract concept, idea or a need may be expressed with many different words. Because of this variety of words usage, it is not possible to have one fixed solution (*i.e.,* results). To address this issue, TechTube implements query normalizing (*e.g.,* stop words

removal, stemming and lemmatizing) and query expansion using pseudo relevance feedback.

Liu et al. [17] proposed a novel hybrid feature location technique that benefits from Pseudo Relevance Feedback (PRF) to improve the performance. Besides the query reformulation using pseudo relevance feedback, Liu et al. [17] used Latent Semantic Indexing (LSI). They used this approach in order to retrieve the results (*i.e.,* information from the source code) as a ranked list of classes, methods, or functions. This approach is named SIngle Trace and Information Retrieval (SITIR). When designing our approach TechTube, we leveraged pseudo relevance feedback to benefit from the first-traced execution of our approach.

Brandt et al. [18] conducted a study with developers to understand their behavior when searching for solutions to a problem. They report that all the study participants (*i.e.,* developers) spent almost 20% of their development time looking for solutions online. Similar to Brandt et al. [18], they observed that the participants refined and augmented their natural language search queries very often. Therefore, in TechTube, we refined the input natural language search queries and augmented them with important keywords retrieved from the available documents using information retrieval methods.

Sadowski et al. [19] investigated developers code search behavior on Google by analyzing 396 responses in a survey and thousands of search sessions. They found that the wideness of the topics (*e.g.,* what, where, why, when, and how does code work) searched by the developers was key to their success in solving the given problems. One third of the time, the input natural language search queries were gradually modified through a query reformulation engine. They also observed that each developer, on average generates 12 natural language search queries per day to address their development issues [19]. Their study also reports the search patterns that developers use to retrieve better results.

Rahman et al. [20] proposed a novel approach called NLP2API that ignites the engine with an input natural language query and extracts the relevant API classes for a development task. In the first step, they provide a set of API classes for a natural language search query from the question and answer threads of Stack Overflow. Then using Borda count and query API semantic proximity, NLP2API extracts the proper classes and retrieves them as a result [20]. Similar to this work, we use pseudo relevance feedback and information retrieval methods to improve the performance and accuracy of the TechTube query reformulation engine.

Carpineto and Romano [21] conduct a survey analyzing the process and outcome of the Query Expansion (QE) using Information Retrieval (IR) techniques. They identify multiple reasons behind the inability of some natural language search query to retrieve the best results. First, the search query is not specifically discussed in a single topic and in most of the cases the keywords used are probably explaining multiple topics. Second, the average size of a

natural language search query for web search is 2.4 words which can be too short, to retrieve appropriate results. Therefore, expanding the search query with multiple meaningful and related keywords can help enhance the performance of retrieved results. Third, after watching an overview of the retrieved results, users increase their previously input search query with multiple different keywords. Considering this fact, utilizing a query reformulation engine that extracts some keywords from primary retrieved results (*i.e.,* using pseudo relevance feedback and information retrieval methods) can help improve the accuracy of the retrieved results. Building on these observations, we implemented a query reformulation mechanism that uses PRF and information retrieval methods.

Rahman and Roy [22] proposed a novel automatic query reformulation approach called –ACER–. They first proposed a novel graph-based term weight approach named–CodeRank– which extracts important terms from source code. afterward, They utilized CodeRank inside their ACER tool to automatically reformulate the input natural language search queries.

Rahman [23] examined search queries used by developers and report that they fail to pick the right structure 88% of the time.

Ksentini et al. [24] discovered that the performance of Information Retrieval systems can be improved by better search queries. To improve search queries, they propose to add keywords from past search queries that provided good results, to a new search query. Although this method is generally considered to be useful, traditional pseudo relevance feedback fails to improve the queries in some cases [25, 26]. For instance, consider some subjects that are irrelevant to each other. The feedback from these irrelevant subjects has nothing in common which causes very bad feedback and eventually adds noisy terms to the structure of the queries. This point has negative impacts on improving the search queries in two ways. First, the search query which was influenced by negative feedback will be replaced and the original query will be ignored in the procedure of query expansion. Secondly, there would not be a relationship between the selected terms and the search query in PRF models [25]. There are many models for PRF, but the most classic and also popular one is Rocchio's model [7, 26]. In this method, the most related terms will be selected and added to the search queries. Also, the relationship between the original query and the top pop-up results is ignored. It is worth mentioning that this model is based on another one called SMART Information Retrieval System which was proposed around the year 1970 [27]. Overall, Rocchio's model is considered to be a very effective relevance feedback method.

Term weighting is another methodology often followed to choose between candidate keywords. Term weighting could be performed based on one of the following methods: (1) frequency-based methods, (2) graph-based methods and (3) probabilistic methods. Among

the aforementioned methods for choosing the relevant keywords, frequency-based methods are reported to achieve better results [28].

There are three main categories of query reformulations techniques, which are explained in the following paragraphs.

Query Expansion: Software developers have to expand their search queries because they hardly ever use the right amount of words in their initial queries. Adding similar or complementary keywords are the most common expansion strategies [23].

Developers often increase the number of keywords in 33%-76% of the queries when they are searching for code on the web [19, 29].

Query Reduction: Search queries often contain noisy, ambiguous or less discriminating keywords. These keywords should be eliminated from the search queries in order to refine them. The keywords that are not specific enough and might lead us to less-related documents should be removed. This method of query reduction is called document coverage [30]. However, some of the significant keywords will not be removed due to weighting methods [31] or POS tagging [32, 33].

Query Replacement: The initial search queries of the developers are not always appropriate, and hence should be replaced [34, 35, 36, 37, 38]. The procedure for replacement could contain spelling corrections [39], generalization or specialization of the query [40, 41].

To expand queries, developers often collect feedback from previously executed queries. This process is often referred to as Query Feedback Collection. There are three major types of Query Feedback Collection techniques.

- **Explicit Relevance Feedback:** This technique relies on the explicit feedback provided by developers about the effectiveness of a query, e.g., a mention that a document was relevant or irrelevant to the information needed [42, 43]. The main limitation of this technique resides in the fact that collecting meaningful and accurate feedback from developers is very challenging and time-consuming.

- **Implicit Relevance Feedback:** This technique relies on implicit information that can be captured through for example eye movements on retrieved documents, document examination patterns, keyword deletion actions, or retention patterns.

- **Pseudo-Relevance Feedback:** This technique relies on information collected automatically (such as the top-K document from a search query) to automatically refine future search results [27, 44, 45].

The collected feedback should be processed for keyword selection. This keyword selection process can be performed through trial-and-error Swanson [46], or using metrics that assess the relevance of each term.

TF-IDF which stands for Term Frequency (TF) - Inverse Document Frequency (IDF) is a metric proposed by Jones [28] in order to specify the relative importance of terms in a text (*e.g.,* news article). TF determine how many time a word appears in a text or document, while IDF is based on the multiplicative inverse of the number of documents in the corpus that contain the target term.

Terms co-occurrence [47] and syntactic dependencies [48] between terms can also be used to select relevant keywords.

Audio Processing

Foote [49] proposed an Automatic Audio Segmentation (AAS) that converts music files to a down sampled mono audio stream. The stream is cleaved into variable sized, non overlapping frames that are beat-synchronized. Then a feature vector is created for each frame and a self-similarity matrix is computed between the vectors. They introduced two thresholds for segmenting the audio flow. One of them is related to a significant novelty score in a 6s interval, and the other one is related to an average amplitude range of the signals for 5 sequential frames [49]. Finally, they employ unsupervised clustering techniques such as Means Of Frames (MOF), Agglomerative Clustering (AC), voting, and Dynamic Time Wrapping (DTW) to label the pre-identified segments.

TechTube leverages audio segmentation to extract chunks between silence points.

Kemp et al. [50] examined the output of three different audio segmenting techniques on news broadcasts. They report that model-based and metric-based segmenting outperform energy-based segmenting. In energy-based segmentation, the power is measured in input signal every 10 ms and smoothed with a 9-frame Finite Impulse Response (FIR) filter. The smoothing process provides the silence part of the audio file [50]. To benefit from model-based segmentation, the input audio is processed by a trained Gaussian Mixture Model (GMM). Since the output of the process is classified audio chunks, they define the segment boundaries as the boundaries between the classified chunks.[50] Metric-based segmentation provides a computed similarity between two neighboring small size windows moving over an audio signal. Hence, the segment boundaries are the points where the calculated similarity is higher than a threshold [50]. In TechTube we apply a mixed energy-based and metric-

based audio segmentation, which measures the average amplitude of audio stream signals and detects the silence signals of the audio from the behaviors of neighboring signals.

Theodorou et al. [51] examined the following three audio segmentation techniques: unsupervised segmentation, data-driven segmentation, and hybrid-model segmentation. At first, they investigate the distance-based (*i.e.,*, unsupervised segmentation) segmentation by measuring the distance between feature vectors. They also analyzed the effect of other audio parameters such as Mel Frequency Cepstral Coefficients (MFCC), Zero Crossing Rate (ZCR), Linear Prediction Coefficients (LPC), and Linear Spectral Pairs (LSP) [51]. [51] report that hybrid techniques which are a mixture of distance-based and model-based algorithms perform better in more complex audio segmentation tasks [51].

Text Processing

Gomaa et al. [52] examined the importance of text similarity measurements in text related search applications such as information retrieval, text classification, document clustering, topic detection, topic tracking, questions generation, question answering, essay scoring, short answer scoring, machine translation, text summarization and others. They suggest 4 different similarity measurement techniques: string-based similarity, corpus-based similarity, knowledge-based similarity, and hybrid similarity measures [52].

String-based similarity techniques measure the similarity of two strings containing a composition of characters [52]. Examples of string based techniques include Longest Common Substring (LCS), Damerau-Levenshtein, Jaro, Jaro-Winkler, Needleman-Wunsch, Smith-Waterman, and N-gram.

Corpus-based similarity technique measures the semantic similarity of multiple words based on the information obtained from a corpus [52]Examples of corpus-based techniques include hyperspace analogue to language (HAL), Latent Semantic Analysis (LSA), Generalized Latent Semantic Analysis (GLSA), Explicit Semantic Analysis (ESA), the Cross-Language Explicit Semantic Analysis (CLESA), Pointwise Mutual Information - Information Retrieval (PMI-IR), Second-order Co-occurrence Pointwise Mutual Information (SCO-PMI), Normalized Google Distance (NGD), and extracting Distributionally Similar Words using Co-occurrence (DISCO).

Knowledge-based similarity techniques utilize the information extracted from semantic networks (*e.g.,* WordNet) to measure the similarity degree between two words.

Hybrid similarity techniques are a mixture of multiple string-based, corpus-based, and knowledge-

based methods. They are reported to achieve better performance in measuring the similarity of multiple texts [52].

Shi et al. [53] proposed ROSF, a novel approach that consists of a mixture of information retrieval and supervised learning, to recommend code snippets with multi-aspect features. They used the BM25 textual similarity method to generate the set of candidates code snippets. BM25 is a textual similarity method that uses the bag-of-words ranking function implemented in Okapi to facilitate the retrieval of documents [53]. BM25 performs well against short queries. Hence, we experimented with it when designing TechTube.

The Semantics of words are very important since dealing with them could mitigate vocabulary mismatch issues in traditional code search engines [17, 54, 55]. This semantic can be captured using the surrounding words in a corpus (*e.g.,* Stack Overflow), according to numerous studies [56, 57, 58, 59].

Word2vec was introduced by Clinchant and Perronnin [60] to capture word associations in a corpus of text.

Ganguly et al. [61] proposed an approach called Generalized Language Model (GLM), which collects word embeddings with query-likelihood language modeling. Ganguly et al. [61] proposed a global approach which is very similar to the Latent Dirichlet Allocation (LDA) of Wei and Croft [62]. In this approach they consider classic global context similarity which is opposite to local term similarity. Like Rekabsaz et al. [63] and Zuccon et al. [64], the authors build on Berger and Lafferty [65] "noisy channel" translation model.

Another approach called Neural Language Translation Model (NLTM) was proposed by Zuccon et al. [64]. In this approach, the authors used word embedding and classic translation model from Berger and Lafferty [65] in query-likelihood IR. When they intended to evaluate the translation probability between terms, they used the cosine similarity of the two terms divided by the sum of the cosine similarities between the translating term and all of the terms in the vocabulary. The important point is that using cosine similarity for translation probability leads to having a positive value in word embeddings. Zuccon et al. [64] used Mutual Information (MI) embeddings for estimating translation probability. TREC datasets AP87-88, WSJ87-92, DOTGOV, and MedTrack reported the performance of the NLTM approach on ad-hoc search. The results of the aforementioned datasets demonstrate that NLTM has a slightly better performance in a large number of topics rather than MI and classic TM systems which experienced large differences on a few topics. Zuccon et al. [64] analyzed various model hyper-parameters in order to induce word embeddings. The analysis indicates that changing the size of embeddings, the size of context window, and model objective (CBOWvs skip-gram) have no straight relation in connection with NLTM's performance vs. baselines.

Zheng and Callan [66] used Deep TR to learn effective query term weights through supervision by constructing feature vectors using word2vec CBOW word embeddings. They constructed each vector with a simple subtraction of the average embedding vectors of query terms from the given term's embedding vector. They performed experiments on ad-hoc search with Indri on 4 TREC test collections: Robust04, WT10t, GOV2, and ClueWeb09-Cat-B, comparing against 3 baselines: BM25, unigram language modeling, and Metzler and Croft [67] sequential dependency model. The results indicate that the performance of DeepTR in queries containing 4 or more words is much higher than the others.

Zamani and Croft [68] proposed two approaches for query expansion which are based on embeddings. These two approaches are very similar to RM1 and RM2 in Lavrenko and Croft [69] relevance models. The first approach which is called EQE1 separates the terms of queries and considers conditional independence between them. However, EQE2 assumes that the semantic similarity between two terms is independent of the query. Another model is proposed by Lavrenko and Croft [69] named embedding-based relevance model (ERM). As in Zamani and Croft [68], ad-hoc search experiments with Galago are reported for TREC AP, Robust04, and GOV2 test collections using keyword TREC title queries, with word embeddings induced via GloVe [70]. Baselines include GLM [61] and VEXP [71]. An unsupervised variant baseline which was proposed by Zheng and Callan [66] is also considered in this approach. The basis of the baseline is how much the vectors are similar to each other and also based on the average embedding vector of all query terms (AWE). The results from PRF compare RM1 and RM2 against ERM. The methods proposed earlier have a better performance than baselines. Also EQE1 tends to outperform EQE2. The authors train three external collection using GloVe and they believe that if different corpora is used to train embedding vectors, the result and values will not be different. In order to produce recognizable similarity values which is appropriate to many IR tasks, the authors proposed that the learning process of embedding vectors should be modified. Moreover, they suggest that they should analyze the way in which they use sigmoid theoretically [68].

Zhang et al. [72] performed a comparison between embeddings collected from Bing query and embeddings learned on a proprietary Web collection. They report that DESM has a poor performance on documents that are considered as large. Consequently, when searching for information using Bing, the collections are first modified and the parts that are not necessary are removed. This is a re-ranking approach that is performed by DESM. Zhang et al. [72] conducted experiments using BM25 and latent semantic analysis (LSA) [73], to compare DESM's IN and OUT embedding space combinations with baseline retrievals. They observed that when re-ranking performed by DESM are utilized, the performance is better in comparison with baselines, especially on the implicit feedback test set. Moreover, when they

train word embeddings in queries and use IN-OUT embedding spaces in document ranking, they have the best performance. According to the Nalisnick et al. [74] hypothesis, the queries that are trained have a better performance because developers tend to include only significant terms from their queries.

Vulić and Moens [75] proposed an approach that is used for monolingual and cross-lingual IR based on word embeddings. As a description of this method, they use the skip-gram model from word2vec to make the corpus learn word embeddings. It is worth mentioning that in this approach, skip-gram is not the only way in which word embeddings are learned, which means that CBOW or GloVe embeddings can be used but they are not discussed in this study.

CHAPTER 3 BACKGROUND

This chapter introduces key concepts used in the research.

3.1 Audio Segmentation

Due to booming music industry, audio segmentation has become popular among developers all over the globe. The developers often segment a music file into different semantic sections (e.g., verse, chorus, bridge [76]). Audio segmenting techniques attempt to identify the change points within an audio file that could be useful for various applications such as monitoring and summarizing a group meeting, and indexing a broadcast news [49, 50, 51].

In the following subsections, we illustrate the most prominent approaches attempting to segment the audio files.

Metric-Based Segmenting

One of these techniques is metric-based audio segmentation. It is quite obvious that there are some times in audio/video in which speakers do not speak, known as speaker's silence.

The metric-based segmenting technique identifies the maximum distance between two neighboring windows and captures a speaker's silence. Then it segments the audio file considering the timestamp of the silence [50].

Sentence-Based Segmenting

The other effective audio segmenting technique adopted to implement TechTube is subtitle-based segmenting. The YouTube videos with a meaningful speech always contain a textual translation of the speakers' speech called subtitle. These subtitles could be uploaded by the authors of the videos beside the videos, or they can be automatically generated using the YouTube speech recognition engine. We extract the available auto-generated subtitle of YouTube technical videos and store them in conjunction with their metadata file. Each of these subtitle files is segmented by analyzing the utterance of textual translations.

In this study, as the best setup of TechTube framework, we exploit the speaker's silence to segment the audio of the video files (Section 4.1.1).

3.2 Query Reformulation

Developers are often searching for code examples. To find relevant code examples, they often use typical queries formulated in natural language texts.

Bajracharya and Lopes [15] report that only 12% of these queries lead to relevant results due to vocabulary mismatch problems [16, 17].

Existing researches [18, 19, 31] indicate that the developers reformulate their queries around 33%-73% of the time. However, they can make mistakes during this procedure and consequently it can cost more time and efforts [31].

Automated query reformulation aids the developers to save their time and more significantly, facilitates their efforts to attain the result they expect. In order to achieve this objective, Adding keywords from a relevant previous result may help the developers mitigate such issues [77].

There have been a number of query expansion techniques using information retrieval methods [21]. We use pseudo-relevance feedback [78] to reformulate a query in TechTube, which has been widely used to expand the search queries [22].

Query Normalization

Before reformulating the input textual query, we first use the query normalizing technique which contains the process of stopwords removal, stemming, and lemmatizing [77].

Term Frequency

TF is defined as the ratio of a word's occurrence in a document to the total number of words in the same document.

$$TF(w, d) = \frac{Occurrence\ of\ word\ (w)\ in\ document\ (d)}{Total\ number\ of\ words\ in\ document\ (d)}$$

Term Frequency - Inverse Document Frequency

This factor measures the importance of a word. Term frequency (TF) does not consider how many times a word appears in different documents. IDF provides a weight to each word based on its frequency in the corpus.

$$TF - IDF(w, d, D) = TF(w, d)\ .\ ln(\frac{Total\ number\ of\ documents\ (N)\ in\ corpus\ (D)}{Number\ of\ documents\ containing\ word\ (w)})$$

Page Rank

Google uses a search engine result ranking algorithm called Page Rank. This algorithm calculates the importance of a web page by counting the number and quality of the web pages.

$$PR(p) = \frac{1-d}{N} + d \sum_{p' \varepsilon M_p} \frac{PR(p')}{L(p')}$$

Where N is the number of pages, M(p) denotes the set of nodes with links to page p, L(p) is the number of outgoing links on page p, and d is the damping factor. The first term of this equation models the possibility that surfers will jump to a random web page (with probability 1 - d). The second term corresponds to the likelihood by which a surfer visits a page following links. Note that the contribution PR(p') from a neighboring page p' is divided by L(p') (the number of p' outgoing links)

In particular, we carefully select the most frequent keywords from the audio contents of technical videos and reformulate a query at hand for retrieving the relevant videos and video summaries (Section 4.2.2).

3.3 Text Similarity

Text similarity measures are popular among researchers and developers due to their high effectiveness in various tasks such as text classification, document clustering, and text summarization [52, 79].

There have been many text-similarity measures, such as string-based similarity, character-based similarity, corpus-based similarity, and term-based similarity [52].

TechTube engine offers text similarity support using both the traditional cosine similarity measure [80, 81] as well as more advanced BM25 algorithm [53]. Cosine similarity computes the cosine value of the angle between two vectorized texts [80, 81].

BM25 algorithm is based on an adaptation of the popular Inverse Document Frequency (IDF) algorithm [82]. Both algorithms use the vectorized representation of the textual contents in the input query and the target documents (that are subject to search) to compute the similarity (Section 4.2.3).

BM25

BM25 is a textual similarity method working based on bag-of-words ranking function implemented in Okapi to facilitate the retrieval of documents [53]

$$BM(D,\ q) = \sum_{t \varepsilon q \bigcap D} IDF(t)\ .\ \frac{tf(t,D)(k_1+1)}{tf(t,D) + k_1(1 - b + b\frac{|D|}{avgdl})}$$

where, *tf(t, D)* is the term frequency of the t in the document D, |D| is the length of document D, and *avgdl* is the average of document lengths in the whole corpus [53].

3.4 Data Collection

We downloaded 98 videos to create the *ground truth* needed for the evaluation of TechTube, and 500 videos for our implemented prototype. We pre-processed all the videos and stored their metadata (e.g., tags, title, description), segmented audio chunks, subtitles, and extracted speech in a database. Our experiment dataset (i.e., all the data except the videos and audio files because of large size) is provided in the replication package for third party reuse and our study replication [8].

CHAPTER 4 TECHTUBE

Figure Figure 4.1 shows the schematic diagram of TechTube.

The framework is composed of an offline component and an online component. The offline component is used to download and preprocess technical videos from online resources.

The online component offers a search and summarization engine of the downloaded videos to a developer. We describe the two components below.

4.1 TechTube Offline Component

The task of TechTube offline component is to download technical videos and their meta-data from online resources. We preprocess the videos into *informative* and *isolated* video segments. These segments are stored for subsequent processing in a dedicated database. TechTube online component uses the database to present the summarized video segments to the developer.

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

4.1.1 Video Chunking

The videos which are downloaded by TechTube offline component will be the input in this step. Likewise, the output is a list of video *chunks*, *i.e.,* sequence of contiguous video frames. One of the practical features of each tutorial video is the lecturer's voice; the descriptive support for the image frames playing sequentially.

Human uses prosody to stress and highlight speech points: silences are an essential prosody feature. If we consider human natural behaviors – more importantly in speaking –, we will unanimously agree that producing silence when we intend to change the subject is inevitable. Speakers, often, put a long silence when they change the subject or underline a very relevant point.

TechTube uses the speaker's silences, the prosody element, to segment utterances. To detect pauses in audio, TechTube uses the amplitude in a given audio file. By amplitude, we mean the value between the middle and the top or the bottom of a sound wave. An amplitude value less than a given threshold is considered silence. Given that different video files can

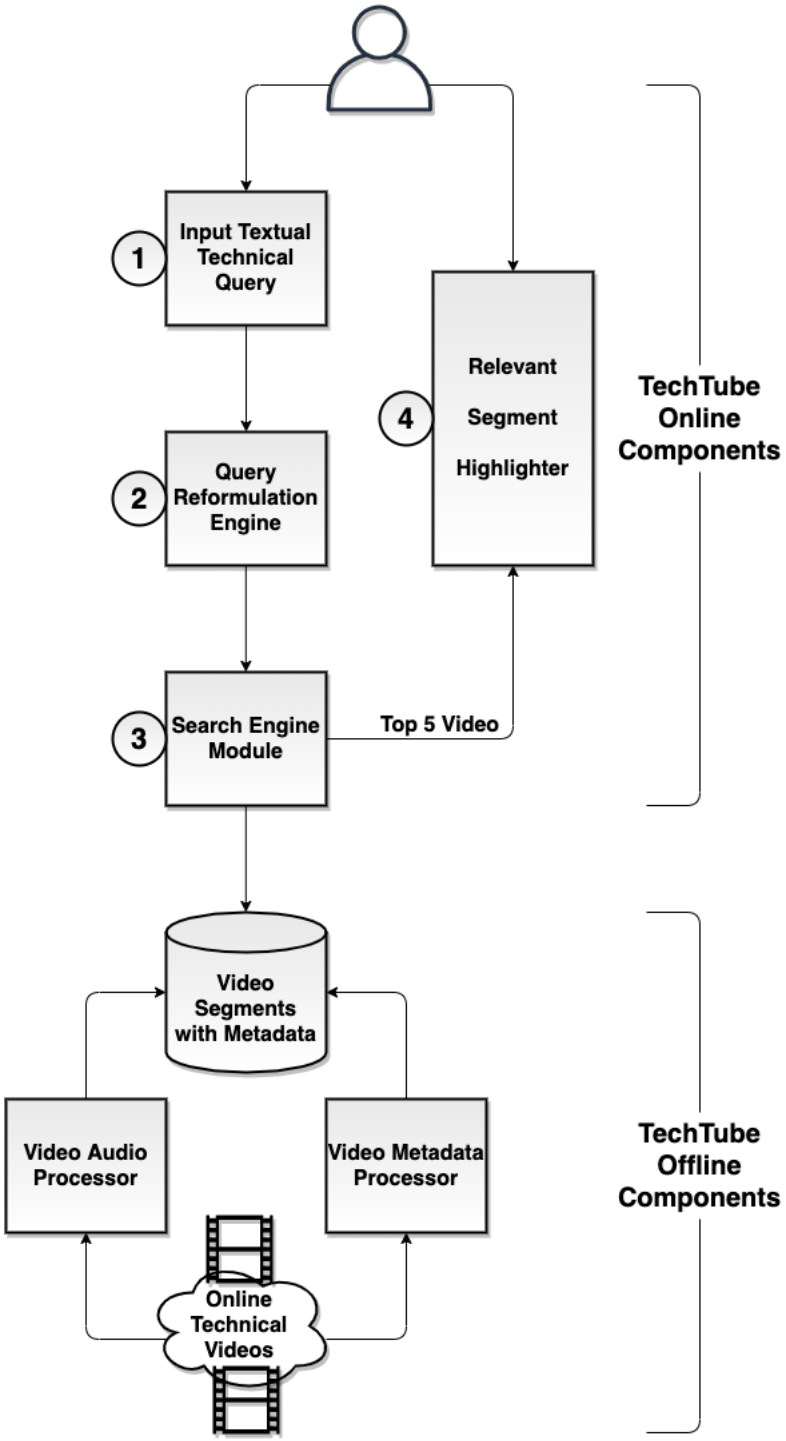

Figure Figure 4.1 The schematic diagram of our proposed TechTube framework

be recorded in different settings, their amplitudes can differ from each other. For instance, a speaker in a tutorial video might have background music, while another may not have any such background music during recording.

Therefore, we dynamically determine the silence per video file as follows. We compute the

amplitude of each frame from the audio of a video. And then we compute the average of the amplitudes. If, for a given frame, the amplitude value drops below the average amplitude value across the frames, we consider that as a silence/pause.

Based on the above detection of silence in the audio, we can then segment an audio file into multiple chunks. That brings us to the fact that, two consecutive chunks are separated by a pause/silence fig Figure 4.2. We create a timestamp for each chunk to specify the starting time and also the finishing time of the chunks based on their time of play on the video.

For video chunking, the TechTube framework currently supports a cross-platform application called FFMPEG [83]. The tool is being applied to the downloaded video files which is considered as an input in order to extract the audio and video parts out of them. To detect silences in the audio, TechTube currently uses PyDub [84].

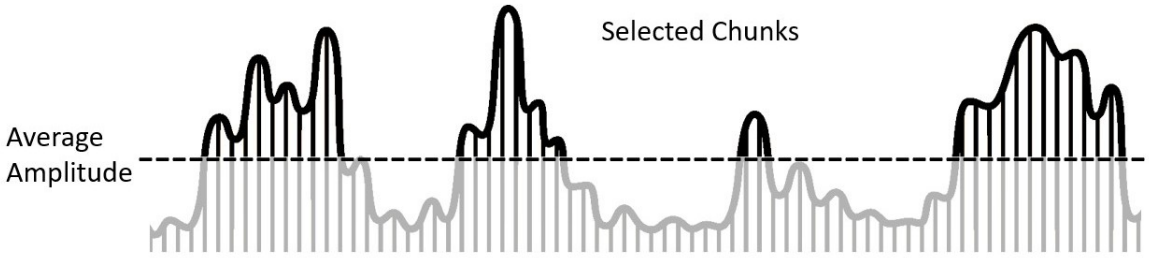

Figure Figure 4.2 Selected video chunks after silence chunks removal

4.1.2 Speech Recognition

The input to this step is a video chunk with its associated audio. The output is a transcribed text out of the audio. TechTube currently uses the Google speech recognition engine [85] to write the speech of each audio chunk into a textual file for similarity comparison and further usages. We selected the Google speech engine because it is a continuous speech recognition engine, it is speaker-independent, plus it scales up to millions of videos. In our manual evaluation of the transcribe texts, we discovered that the recognition accuracy of the engine was 90% (average).

4.2 TechTube Online Component

In the previous section, we described how we segment the audio and extract texts out of the segmented chunks. The online component provides an interface for searching the downloaded videos using a natural language query.

Given as input a textual query related to a technical task, TechTube utilizes a query reformulation method to discover the appropriate videos by mining their metadata, such as tags and video descriptions (if applicable). Finally, TechTube measures the similarity rate of the reformulated query and the extracted speech texts by using the text-similarity algorithms. For each of the top query results that are initiated by the developers, TechTube then visualizes the relevant fragment of the video in a summarized interface. Specifically, the online component is composed of four modules:

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

We describe the four modules in detail below.

4.2.1 Input Textual Technical Query

TechTube allows a developer to describe a technical task using natural language in a search box. One of the main parts of search tasks that impact the relevancy rate of the results is the generated search query. A well-dictated non-noisy rich query leads to gaining more accurate outcomes. Due to this issue, TechTube uses a necessary query cleaning and pre-processing method for text normalization, which is a combination of stemming, lemmatization, and stopword removal [77].

4.2.2 Query Reformulation Engine

The textual description of a task (i.e., the query) provided by a developer may not contain all the necessary keywords to describe the task properly.

To address this problem, TechTube reformulates the query using a standard query expansion technique – Rocchio's expansion [7]. TechTube uses the technique as follows. Each downloaded video in our database contains metadata (e.g., title, tags, description) besides their

subtitles (using auto-generated subtitles from YouTube) and audio speech that explains its discussion topics.

Our approach first captures pseudo-relevance feedback [27] on a developer's query by executing the query against all the available videos. We pick the topmost retrieved videos from the query results. The underlying idea is that these somewhat relevant videos might contain essential keywords that could complement the developer's free-form query.

TechTube then finds all the keywords in the feedback videos that we collected using the TechTube offline component. It then picks the most frequent keywords from these feedback videos. These keywords are then used to expand/reformulate the search query issued by the developer. A similar approach has been adopted by many relevant studies from the literature [22, 42, 86].

4.2.3 Search Engine Module

There would be two tasks for which the Search Engine Module is responsible. This module is responsible for two tasks. First, it detects the four most relevant videos given as input the reformulated query. Second, for each returned relevant video, the module further detects chunks in the video that are relevant to the query.

The search engine module utilizes Apache Lucene [87]. We selected Apache Lucene for two reasons. Apache Lucene is a popular open-source search engine software library and also it is regularly used in state of the art query reformulation research in Software Engineering (see [88]). To compute the similarity between the input query and a video, TechTube takes into account the speech text of the video.

TechTube produces a vectorized representation of the input query and the speech text. To produce the vectors, TechTube applies standard text normalization approaches, such as stopword removal and lemmatization.

The search module offers two similarity metrics to compute the similarity between the input query and the speech text: Cosine Similarity [82] and BM25 [53]. Upon computation of the similarity, TechTube sorts the videos by the similarity values. TechTube takes the top four videos with the most similarity values. For each of the five videos, TechTube then computes the similarity between the input query and each chunk in the video.

4.2.4 Relevant Segment Highlighter

At this stage, for each relevant video identified by the Search Engine Module, TechTube has several, non-contiguous, sequences of similar chunks. To ensure a coherent and smooth documentation experience from the video given the input query, TechTube needs to combine similar chunks. However, it may happen that two very small relevant chunks are separated by a pause. In such cases, the chunks may be relevant, but they cannot provide a coherent experience. Therefore, to elect the most relevant video fragment (*i.e.,* non-contiguous chunks sequence), TechTube uses the Longest Common Subsequences (a.k.a., LCS) [89] strategy. Figure Figure 4.3 explains the strategy as follows.

The black bars are relevant video chunks. The white bars are less relevant video chunks and the spaces between sequences denote pauses in the audio. Each bar denotes a video chunk. The length of a bar corresponds to its similarity to the input query, i.e., a longer bar is more relevant.

To produce a coherent representation of the relevant video chunks out of an input video, TechTube starts with the first video chunk that is similar to the query. For reducing the noise, TechTube only considers a video chunk with a similarity higher than a threshold (half similarity of the highest similar chunk). TechTube then proceeds with the next video chunk. If the chunk is relevant, it is included. If several consecutive chunks are found to be relevant, they form a sequence.

This approach stops when a chunk is found to have a pause, or its relevance measure is lower than the threshold. For example, in Figure Figure 4.3, the first LCS has a total of seven consecutive video chunks. The second LCS has a total of six consecutive video chunks. To offer a uniform and coherent documentation experience out of the video, TechTube picks all the video chunks between LCS1 and LCS2 (i.e., all including the pauses in between).

Using this same approach, the two consecutive relevant video chunks after LCS2 are not included, because they are significantly smaller in length than the previous two longest common subsequences (i.e., 7 for LCS1 and 6 for LCS2).

We apply this process to each of the five selected videos. Each video automatically starts at the beginning of the first LCS in the video. Upon the finish of the first LCS, the user is taken to the second LCS of the video. If a user wants to watch the original videos, they can ignore the timing of the video and start the video manually from the beginning time or any other time.

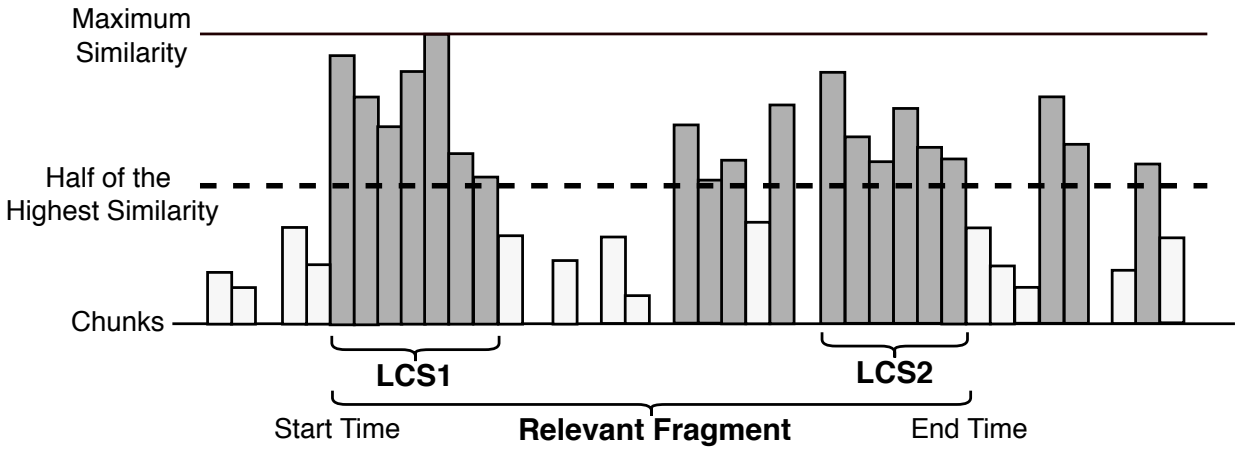

Figure Figure 4.3 TechTube Relevant Fragment Identification

4.3 TechTube Web-Based Prototype

Our current prototype of TechTube provides a web-based interface. In Figure Figure 4.4, we present a screenshot of TechTube prototype. TechTube allows users to input their search queries ① and submit them by pressing a search button ②. Then TechTube navigates the users to the result page. In the result page, the TechTube provides users top five related videos ③ to the search query. For each video, the start time and end time ④ of the relevant fragment are available under the video box. By pressing the play button on the video frame, The video starts at the appropriate fragment start time and finishes automatically at the end time of the relevant fragment. The users always access the search box ⑤ to search for new queries.

The current implementation of TechTube uses videos from YouTube. To download the videos, we use a command-line based tool called YouTube-dl. Besides that, we extracted the meta-data of the videos for their helpful information such as video YouTube ID (URL), tags, and descriptions.

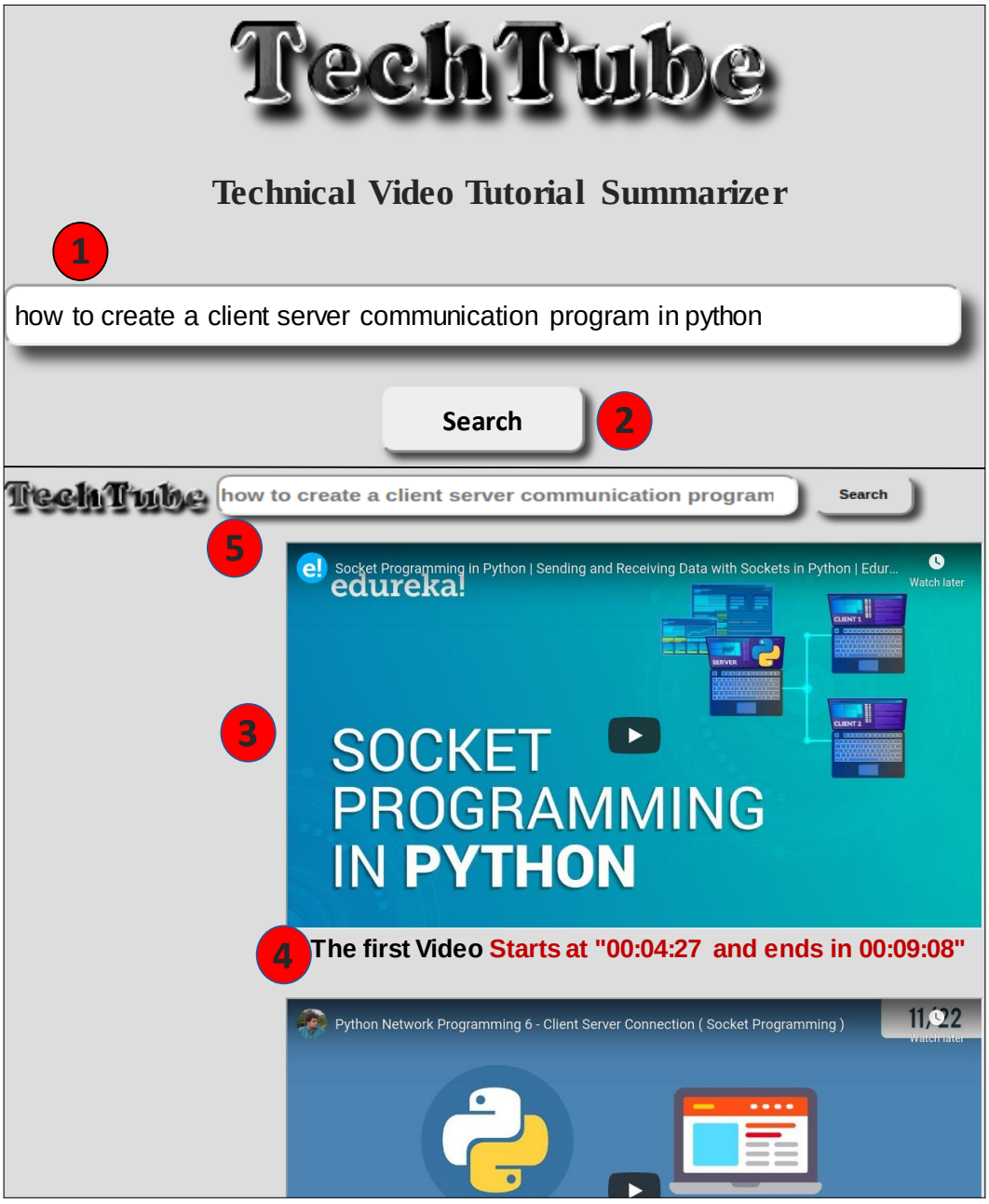

Figure Figure 4.4 TechTube Prototype as a Web-based Search Engine

CHAPTER 5 EXPERIMENTS

We evaluate TechTube in two different ways: (1) by means of an empirical evaluation and (2) by a user study, which is a case study that involves 16 developers.

The first empirical evaluation focuses on TechTube's effectiveness to retrieve relevant videos and to produce meaningful video summaries. In particular, we examine the roles of three TechTube components, namely – retrieval technique used in the Search Engine Module (*e.g.,* Apache Lucene), video splitting method, and query reformulation mechanism.

In a second step, we conduct a user study involving 16 professional developers and investigate whether TechTube provides relevant, concise, useful video summaries against their search queries. Overall, we address five research questions. While RQ_1 to RQ_4 deal with our empirical evaluation, RQ_5 concentrates on the developer study.

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

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

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

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

RQ5. Does TechTube deliver relevant, concise, useful video content against the queries of developers?

5.1 Experimental Dataset

We collect a total of 98 natural language queries related to Online Repository (e.g. Github) maintenance, Java and Python programming tasks for our experiments due to their popularity among developers. The queries related to Java programming were taken from an *existing benchmark* [20] whereas the rest of the queries were extracted from the top-scored Q&A threads of Stack Overflow considering the post topic as the query. We focus on these fields because of the variety of the videos and their popularity among developers and software engineers. We then populate our database by downloading 400 technical videos from YouTube that were retrieved using the keywords from these 98 queries and that looked relevant to the queries at the first glance. Our inclusion criteria were: (1) each video should be longer than three minutes (three minutes videos can be considered as a video summary), and (2) the video should contain an English speech track. Since some of our queries were similar in the concept, we could use multiple queries against one video, so we extract 98 videos out of 400 videos to have one video per query where those videos present different types of technical tutoring such as development, code explanation, package installation, repository maintenance and etc.

In summary, the experimental dataset encompasses different types of videos; different by time duration, visual quality (*i.e.,* low quality frames, high quality frames) style (*i.e.,* slide presentation frames, code development frames and etc.), single and multiple topics (*i.e.,* videos that provides multiple topics the same video).

We also collect the metadata (*e.g.,* tags, title, description) and the subtitles (*i.e.,* textual narrations generated automatically by YouTube) for each of the videos for our experiments.

We construct a *ground truth* (*i.e.,* relevant videos plus segmented relevant video sections) database for the 98 queries as follows. We first segment each video manually. Then three software engineering graduate students who did not have any clues about the approach manually analyzed the videos and the segments to assess their relevancy to the 98 queries. Segmenting was organized in two activities. In a first activity, each participant segmented and labeled the same 10 videos. Following this step, participants compared segmentations build a consensus and agreed on a segmentation and labeling strategy. In the second activity, once the common strategy was achieved, each individual participant labeled his share of dataset videos. About 50 man-hours were spent to construct the ground-truth. We use this ground truth database to answer RQ_1 – RQ_4.

Our experimental dataset and replication package can be found online [8] for the replication and third-party reuse.

5.2 Performance Metrics

TechTube's performance is evaluated using six metrics. The first three metrics, Hit@K, MRR and MAP are used to evaluate the retrieval of relevant videos. The other three metrics, precision, recall and F1-score are used to evaluate the retrieval of relevant video sections. We describe each metric below.

Hit@K

Hit@K calculates the percentage of queries for each of which at least one ground truth video is retrieved within the Top-K results.

$$Hit@K(Q) = \frac{\sum_{q \in Q} isCorrect(q, K)}{|Q|}\%$$

Here, *isCorrect(q, K)* returns a value of either 1 or 0. If the ground truth video for the query q exists within the Top-K ranked list, then the output is 1, and otherwise 0. Q represents the set of all queries used in the experiment.

Mean Reciprocal Rank (MRR)

Reciprocal Rank is the multiplicative inverse of the position of the first *ground truth* video within the Top-K results for a query. MRR averages this measure for all search queries Q as follows.

$$MRR(Q) = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{rank(q, K)}$$

Here, *rank(q, K)* returns the rank of the first relevant video within the ranked list of size K. If the relevant video is retrieved as the first result, *rank(q, K)* returns 1. If the video is not found within the list, the function returns a rank of ∞.

Mean Average Precision (MAP)

Average Precision (AP) is the average of the precision@k calculated at the occurrence of every *ground truth* video within the ranked list for a query q. MAP is the mean of average precision for all search queries Q.

$$AP = \frac{\sum_{k=1}^{K} P_k \times rel_k}{|RR|}$$

$$MAP(Q) = \frac{\sum_{q \in Q} AP@K(q)}{|Q|}$$

Here, rel_k returns 1 for a relevant result at k^{th} position, and 0 for an irrelevant one. P_k calculates the precision at the k^{th} result (a.k.a., precision@k) that returns a value between 0 to 1. RR stands for set of relevant results retrieved by a query.

Precision

Precision calculates the fraction of retrieved video sections that overlap with the ground truth video sections for a query.

$$Precision(Q) = \sum_{q \in Q} \frac{GT_q \cap TT_q}{|TT_q|}$$

Here, $|\mathrm{GT}_q \cap TT_q|$ returns the overlap time between the *ground truth* video summary and summarized video from TechTube for a query q. $|TT_q|$ returns total duration of the summarized video from our tool. Precision ranges between 0 to 1.

Recall

Recall calculates the fraction of ground truth video sections that are retrieved by an approach for a query.

$$Recall(Q) = \sum_{q \in Q} \frac{GT_q \cap TT_q}{|GT_q|}$$

Here, $|\mathrm{GT}_q \cap TT_q|$ returns the overlap time between *ground truth* video summary and summarized video from TechTube for a query q. $|GT_q|$ returns total duration of the *tround truth* video summary. Recall ranges between 0 to 1.

F1-Score

F1-Score is a harmonic mean of precision and recall for each query. It is also averaged over all queries as follows.

$$F1 - Score(q) = \sum_{q \in Q} 2 \times \frac{Precision(q) \times Recall(q)}{Precision(q) + Recall(q)}$$

Here, F1-Score is a combined metric for evaluating the precision and recall. It returns a value between 0 to 1.

5.3 RQ_1: Performance of TechTube in Retrieving Relevant Video and Relevant Video Segments

First, we assess the performance of TechTube to retrieve relevant videos given for an input query. We compute the similarity of the relevant videos utilizing the BM25 metric in TechTube search engine module.

Second, we evaluate the performance of TechTube to retrieve pertinent video segments from a video and compare the output with CodeTube [4], a closely related existing approach.

We run each of our 98 queries against TechTube. For each query, we pick the top-5 results by keeping their ranks as returned by TechTube and then compare the results against our ground truth database using three performance metrics – Hit@K, MAP and MRR. Table Table 5.1 thoroughly portrays the results. Immediately following relevant videos are retrieved, we also extract the relevant video sections (a.k.a., video summary) using both approaches (i.e. TechTube and CodeTube).

We compare the relevant video segments of each approach against our *ground truth* database by computing three other metrics –precision, recall, and F-score. Fig. Figure 5.1 reports the obtained results.

Table Table 5.1 TechTube's Performance in Video Retrieval

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

Table Table 5.1, shows that TechTube retrieves the relevant videos for 93% of the 98 search queries within their Top-5 result. The mean average precision is 76%. This fact demonstrates that TechTube retrieves a relevant technical video for 9 out of 10 queries. It also achieves a mean reciprocal rank of 0.76, which indicates that, on average, the most relevant videos can be discovered at the first or second position within the ranked retrieved results. Table Table 5.1 shows that TechTube delivers the most relevant videos at the topmost position (*i.e.,* Hit@1) for 64% of the queries.

Fig.Figure 5.1 shows that, on average, TechTube can identify the most relevant video fragments for a search query with a precision of 67%, a recall of 53% and an F-score of 50%.

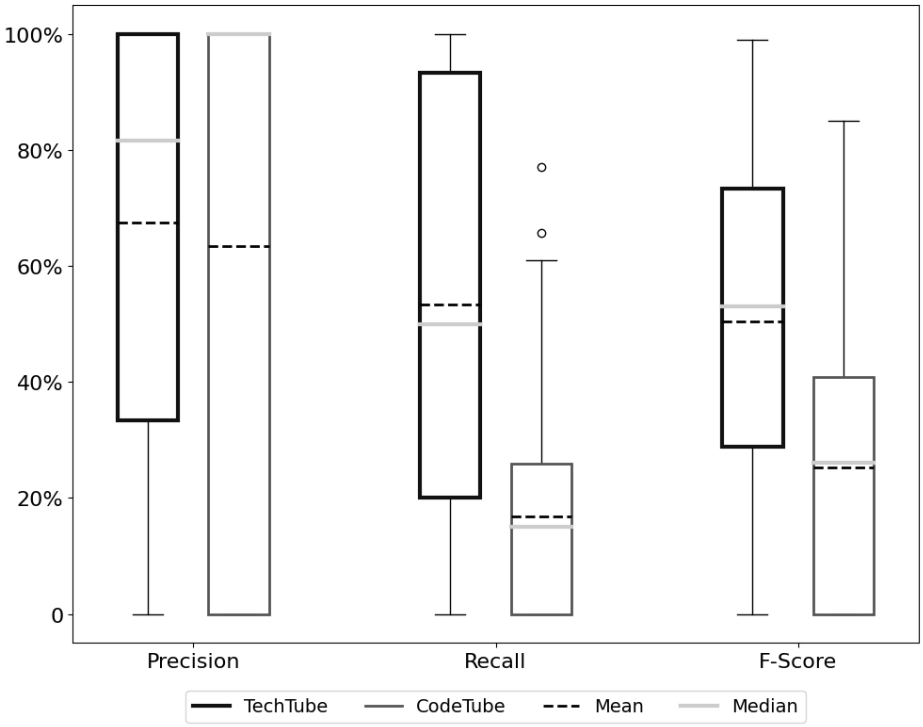

Figure Figure 5.1 Comparison between TechTube's performance and CodeTube's performance in the retrieval of relevant video sections

Table Table 5.2 shows the detail of the comparison between TechTube and CodeTube approaches against a variety of videos such as videos related to Java programming, Python programming and videos with long duration (i.e. more than 15 minutes) and the videos with a duration of less than 15 minutes.

On Table Table 5.2 we notice that, TechTube delivers relevant sections for the Python related video tutorials with 62% precision which outperforms the CodeTube approach by 13% higher precision, 180% higher recall, and 140% higher F1-score. Due to the use of Java based filtering and the use of island parser to find the Java constructors, CodeTube provides a weaker performance on Python related video tutorials.

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

Besides, on long duration videos (i.e. videos having more than 15 minutes of time duration) TechTube achieves 11% more precision, 41% more recall and 25% more F-score compare to the CodeTube approach. Comparison of both approaches on table Table 5.2 shows that the TechTube video summaries retrieved with 7% more precision on videos in Python programming topic.

By comparison with CodeTube, it can identify the most relevant video section for a search query with a precision of 64%, a recall of 17% and an F-score of 25%. In total, our approach – TechTube – outperforms the CodeTube approach by 6% higher precision, 230% higher recall, and 100% higher F1-Score.

CodeTube approach mostly focuses on Java programming video tutorials and uses Java keywords filtering on the OCR content and uses an island parser to identify the Java constructors in a tutorial. Ponzanelli et al. [4] approach still works on Python programming tutorials and provides 55% precision while TechTube boosts up to 62% precision.

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

Indeed since TechTube does not rely on visual features (e.g. video frames, OCR extracted content and etc) and only analyzes the speech and audio of the video files and measures the textual similarities, it performs better in a variety of video types with different time duration and different visual features against other existing approaches. Because of video frame noises (e.g. low quality video, zooming in and zooming out of a frame by the tutor) and a Java dictionary-based filtering of the OCR extracted information, CodeTube provides less precise video segments against our experimental dataset.

Summary of RQ_1: TechTube can retrieve relevant technical videos for **93%** of the queries with **76%** precision. It also can deliver the relevant sections from these videos (a.k.a., video summary) with 67% precision which is 6% higher than CodeTube, 53% recall which is more than 200% higher than CodeTube, and 50% F1-Score which is 100% higher than the CodeTube approach.

About 70% of the summarized videos overlap with the ground truth video summaries. We also check the distribution of these performance metrics with quartile analysis. TechTube achieves a median precision of 82% (*i.e.,* Q2), which is highly promising. While it can deliver

about 50% of the ground truth video fragments for half of the queries, almost the whole video summary (*i.e.,* 95%) is delivered for 25% of the 98 queries.

Table Table 5.2 TechTube's Performance in Video Section Retrieval

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

5.4 RQ_2: Impact of Video Search Engine in TechTube

TechTube employs a search engine module (Step 3, Fig. Figure 4.1) accepting as input a natural language query and returning relevant videos.

We use two different methods – *ad hoc* and *Apache Lucene* – for retrieving the videos where each video document is represented with its metadata (*e.g.,* tags) and subtitles collected from YouTube. The ad hoc method relies on *cosine similarity* algorithm [81] whereas Lucene employs a sophisticated Information Retrieval technique called BM25 [53].

We experiment with both search engines – ad hoc and Lucene, and compare their performance in the relevant video retrieval using three performance metrics – Hit@K, MAP and MRR. Table Table 5.3 summarizes our comparative analysis. We thus answer RQ_2 as follows.

Table Table 5.3 Role of Search Engine in Relevant Video Retrieval

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

ST = Subtitles, **T** = Tags

Table Table 5.3, demonstrates the perfect performance of both search engines –ad hoc and Lucene– in retrieving the relevant videos. We represent each video document as different combinations of its tags and subtitles and then investigate how these two search engines can retrieve the relevant technical videos.

When subtitles are used as a proxy to video documents, both search engines achieve about 83% Hit@5 with ≈72% mean average precision, which is quite promising.

When tags are used as a proxy to video documents, the ad hoc method achieves 17% higher

Hit@5, MAP and MRR than those of Lucene. Since tags contain only a few keywords, they are better suited for the ad hoc keyword search than the Lucene.

However, when both subtitles and tags are combined and used as a proxy to video documents, Apache Lucene achieves the best performance in retrieving the relevant videos. The BM25-based search engine (a.k.a., Lucene) delivers relevant videos for 93% of the search queries with 76% mean average precision and 0.76 mean reciprocal rank, which is highly promising.

Such findings justify our choice of using Lucene as the search engine of TechTube. It also should be noted that TechTube is not restricted to Lucene and thus can be easily extended with other available search engines (*e.g.,* YouTube search API, Indri).

Summary of RQ_2: Apache Lucene achieves 93% Hit@5 with 76% precision in retrieving the relevant technical videos and outperforms an ad-hoc method, which **justifies** our choice of using Lucene as the search engine.

The ad-hoc search engine finds the relevant video 92% of the time, within the Top-5 retrieved videos as results with a mean average precision of 75%. The ad-hoc reciprocal rank is 0.75, which means that the ad-hoc search engine finds the most relevant video as the first or second retrieved results.

Unlike ad-hoc, Lucene search process the combined metadata files and subtitles file better and improve the ad-hoc video retrieval technique by the aspect of retrieval accuracy, average precision, and reciprocal rank. The Lucene search engine finds the relevant video 93% of the time within the Top-5 retrieved results with a mean average precision of 76% and a mean reciprocal rank of 0.76 as well which means that the Lucene also retrieves the relevant video as the first or second position of the Top-5 retrieved videos. TableTable 5.3 presents that the Lucene search engine and ad-hoc search engines retrieve the video with similar accuracy while using only subtitle files. Hence the ad-hoc technique provides 6% higher average precision and reciprocal rank.

The comparison shows that the ad-hoc method beats the Lucene mechanism dealing with only metadata files as the source of searching. However, in total, the highest Top-K Accuracy, Mean Average Precision@K, and Mean Reciprocal Rank@K belongs to the Lucene search engine when using the mixed metadata and subtitles files. TableTable 5.3 demonstrates that the Lucene search engine improves the best setup of the ad-hoc search engine for 1% in Top-5 accuracy, 1% Mean Average Precision@5 and 0.01 Mean Reciprocal Rank@5.

5.5 RQ_3: Speaker's Silence as the Video Splitting Method

The silence detection and splitting mechanism could be considered as one of the distinguishing characteristics of TechTube. At the video chunking step (Section 4.1.1) each video is divided into multiple sections, which is absolutely fundamental to identify the relevant video sections. We employ two video splitting methods –*speaker's silence* and *subtitle sentence*.

The silence-based method splits a video into multiple sections based on the speaker's silence as explained in 4. Following silence detection, we run Google Speech Recognition Engine [85] to convert the vocal (and non-vocal) sound into the textual contents. Such a finely chopping process should be performed on the videos from which collected YouTube with the assistance of their subtitles. The subtitle sentence-based method splits each video using the duration of each sentence from these video subtitles.

We experiment with silence-based splitting and subtitle-based splitting methods and determine the performance of TechTube in retrieving the relevant video sections using three performance metrics – precision, recall, and F1-score. Fig. Figure 5.2 summarizes our comparative analysis.

Table Table 5.4 Evaluation of silence-based segmented speech against the subtitle-based segmented speech

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

From Fig. Figure 5.2, we can recognize that the precision of retrieving relevant video fragments from the technical videos would be 68% on average in the TechTube, when the silence-based splitting method is used. On the contrary, such precision is about 64% with the subtitle-based splitting method which demonstrates a weaker performance in comparison.

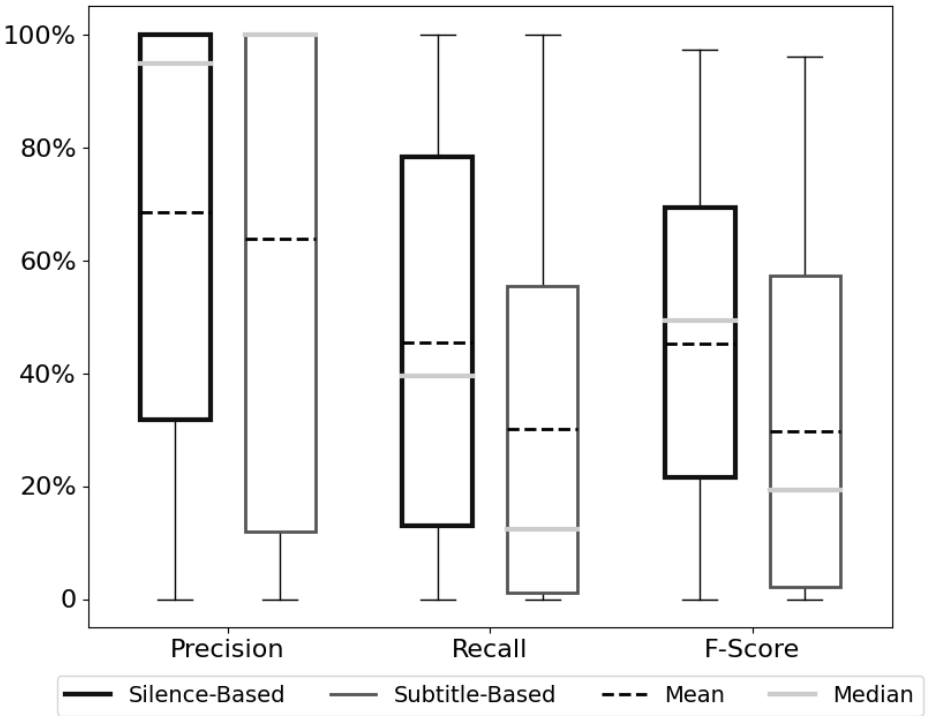

Figure Figure 5.2 Comparison between silence-based video splitting and subtitle-based video splitting for the retrieval of relevant video sections

However, TechTube achieves 51% higher recall and 53% higher F1-score in retrieving relevant video sections with the silence-based method in comparison with the subtitle-based method.

From the box plots in Fig. Figure 5.2, we also notice that the mean and median measures for the silence-based method are comparatively higher than those for the subtitle-based method. All these findings elucidated above justify our choice of using speaker's silence as the video splitting method of TechTube.

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

Unlike prior technique, segmenting the subtitle by using line-by-line strategy finds the relevant fragment of the videos by a mean precision of 64%, recall of 30%, and an F-score of 29% on average. The comparison between the two mentioned segmenting techniques reports that the silence-based segmenting technique, on average, ameliorates the mean precision, recall,

and F-score for 4%, 15%, and 16%, respectively.

5.6 RQ_4: Benefits of the Query Reformulation Method

To improve results, TechTube employs a query reformulation engine (Section 4.2.2) that complements these queries with important keywords from the video speeches. For this purpose, in particular, we select the most frequent keywords from the speeches of relevance feedback videos for query expansion (details in Section 4.2.2). They might have a better chance of identifying the relevant technical videos and the relevant sections from these videos.

In order to reformulate the input textual query, we consider using different pseudo relevance feedback techniques. Accordingly, we compared Term Frequency (a.k.a, TF), Page Rank (a.k.a., PR), and Term Frequency - Inverse Document Frequency (a.k.a., TF-IDF) on our experimental dataset to provide TechTube with a well-suited query reformulation method.

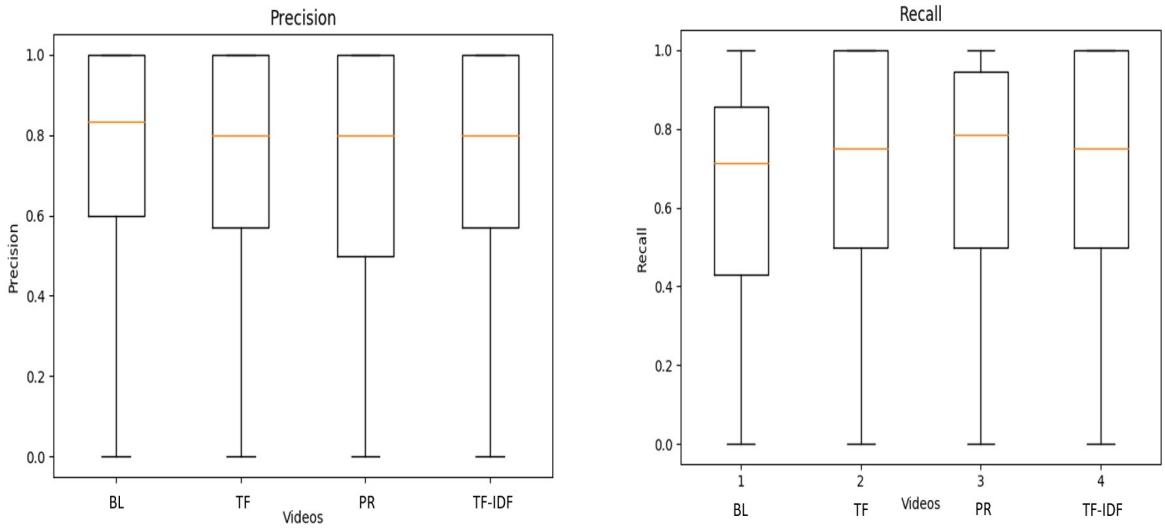

Figure Figure 5.3 Comparison between different query reformulation techniques

Fig Figure 5.3 shows the comparison between different methods of query reformulation based on information retrieval techniques. The result of the query reformulation methods is very similar in terms of precision and recall, were close. Due to the higher rate of accuracy in retrieving the results, we consider using TF as part of the query reformulation setup.

In our experiment, we investigate whether such a query reformulation improves TechTube's performance or not. We thus experiment with two types of queries – free-form, user provided queries (a.k.a., baseline queries) and reformulated queries, and determine how our approach – TechTube – operates in retrieving the relevant videos and relevant sections from them. Table Table 5.5 and Figures Figure 5.4, Figure 5.5 summarize our comparative analysis.

Table Table 5.5 Role of Search Queries in Relevant Video Retrieval

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

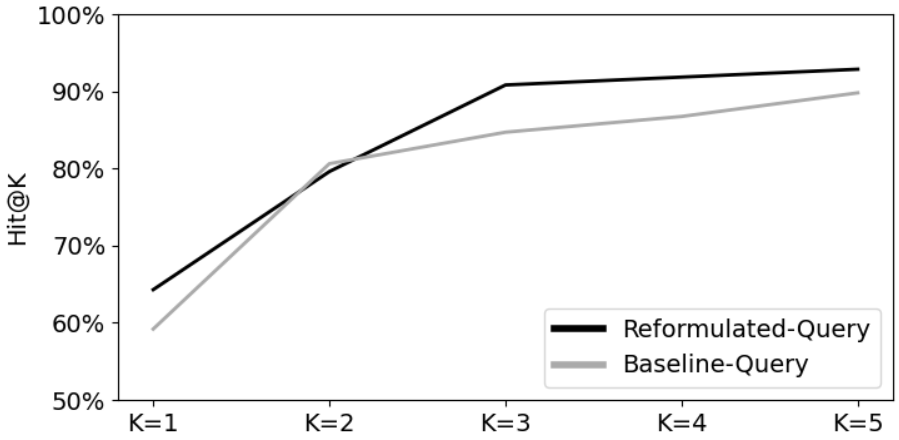

Figure Figure 5.4 Comparison between baseline and reformulated search queries in the relevant video retrieval

RQ_4 (a)-Impact of Query Reformulation in the Retrieval of Relevant Videos:

From Table Table 5.5, we realize that the percentage of retrieving the relevant videos in TechTube, using the baseline query, is 90% of the queries within the Top-5 results. with a mean average precision of 72% and a mean reciprocal rank of 0.72. However, analyzing the performance indicates that the reformulated search query improves the result precision.

Our approach delivers the relevant videos 93% of the time within the Top-5 results when reformulated search queries are applied by Apache Lucene. It provides these relevant videos with a mean average precision of 76%, and a mean reciprocal rank of 0.76, which are 5% higher. Furthermore, the reformulated queries achieve 7% higher Hit@3 than the baseline queries in retrieving the relevant videos.

We also notice that reformulated queries outperform the baseline queries when ad-hoc search engine is used. From Fig. Figure 5.4, we also see that the reformulated queries outperform the baseline queries for various Hit@K measures when they are executed with Apache Lucene. All these findings clearly suggest that the positive impact of query reformulation on TechTube's performance.

RQ_4 (b)-Impact of Query Reformulation in the Retrieval of Relevant Video Fragments:

Figure Figure 5.5 shows the comparison between baseline query and reformulated query in delivering the relevant sections from the technical videos. It is evident that TechTube, on average, achieves 67% precision with both baseline and reformulated queries. That is, ≈70% of the retrieved video sections overlap with the ground truth. However, our approach achieves 15% higher recall and 9% higher F1-score with the reformulated query than with the baseline query.

Table Table 5.6 Evaluation of reformulated query against the baseline query for relevant section retrieval

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

Overall, the query reformulation mechanism enhances TechTube's performance in retrieving the relevant sections from the technical videos.

Summary of RQ_4: Query reformulation **complements** the free-form, user provided search queries and improves TechTube's performance in the retrieval of relevant technical videos and relevant video sections from them, which **reminds us with an acceptable and reasonable explanation** of choosing the query reformulation.

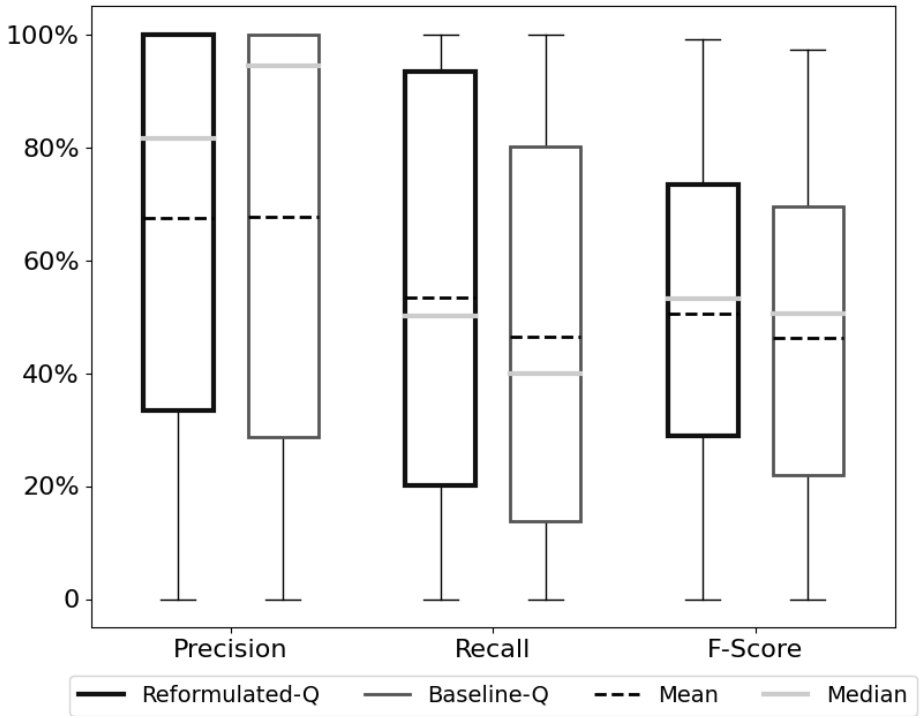

Figure Figure 5.5 Comparison between baseline query and reformulated query in the retrieval of relevant video sections

TechTube retrieves the relevant fragment of the videos to a given baseline query with a precision of 67% compare to the *ground truth*. Furthermore, Using baseline search query in video relevant fragment retrieval, TechTube, on average, provides the retrieved video summary with a recall of 46% and an F-score of 46% as well.

Besides the performance of TechTube for baseline search queries, our approach, on average, retrieves the video relevant fragment to a reformulated search query with a precision of 46%, a recall of 53%, and an F-score of 50%.

The comparison between the results of the metrics for both types of search queries reports that the TechTube approach performs better while uses reformulated search queries.

Despite the same precision and higher median of the retrieved fragments for the baseline search queries, reformulating the search queries using the Term Frequency method on average boosts the recall of the retrieved relevant fragments by 7%, and also improves the F-score of the retrieved fragments for 4%.

5.7 RQ_5: Evaluation of TechTube with Developer Study

In this study, a total of 16 developers participated in the study. Each developer completed six different programming tasks using TechTube.

We manually checked each completed task for accuracy. After completing the tasks, each user was invited to provide their feedback on their overall experience of using TechTube. Besides, in the second phase of the study, we compare the summarized videos against the original videos using an online survey. In the remainder of the section, we briefly explain the study setting and the results.

TechTube Setup

We create a web-based prototype of TechTube from which users can access online. Almost 500 video tutorials related to six programming tasks (Table Table 5.7) are provided from YouTube. After collecting it is time to pre-process those videos using TechTube's offline component.

Participants

We recruit 16 developers with more than one year of programming experience in one or more of the following languages: Python, Java, C#, Javascript, and PHP. As you might know, all of these languages are among the most popular programming languages in Stack Overflow, one of the most popular Q&A sites for developers all over the globe.

In the meantime, such a variety of programming languages also ensures that we can reliably determine the effectiveness of TechTube across developers of diverse expertise. Furthermore, such variety guarantees that the TechTube approach performs well against various types of videos apart from the programming language explained as the content.

Out of the 16 participants, nine developers were graduate students while seven of them professionally worked as developers.

The professional developers are found via the online site Freelancer.com. We made a contact with all developers directly (e.g. via email and chat) and delineated the whole study in detail.

Overall, 16 developers with different preferences for programming languages participated in our study. Fourteen of them have more than one year of experience in programming with Python.

Table Table 5.7 Programming tasks selected for the developer study

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

Study Setup

We choose six tasks based on two criteria (see Table Table 5.7). First of all, five programming languages were chosen to suit the expertise of the study participants: Python, Java, C#, PHP and Javascript. Second, the tasks are in different levels in terms of difficulty whereas they should not be tedious and should not require a long time (*e.g.,* more than 30 minutes).

In the first phase, we perform a pilot study which can show us determine the difficulty level and the average implementation time of the tasks. Based on the pilot study, we divide the six tasks into three groups – easy tasks (*e.g.,* Task 1, Task 2), moderate tasks (*e.g.,* Task 3, Task 4), and difficult tasks (*e.g.,* Task 5, Task 6). Accordingly, we consider having a balanced distribution of the task among the participants.

Based on a pilot study, each of these tasks takes from 20 to 30 minutes to complete. In order to attain comparable results, we provide each participant with a pre-configured virtual machine (VM) equipped with TechTube and the required tooling for implementing their tasks (*e.g.,* IDE).

All activities of a developer are recorded by the virtual machine. Based on such VMs and videos they record, we identify the search queries issued by the developers for each of the tasks. Besides the queries, we also identify the video summaries generated for them and what they used as guides in the completion of the tasks.

In the second phase, we aim to evaluate the relevance of the video summaries (generated by TechTube) against the developer queries issued during their tasks. We thus extract the frequently used queries for each task and the frequently retrieved video summaries for them to conduct an online survey.

Since almost all of our participants have more than one year programming experience in Python, we collect the queries related to Python programming tasks. In our survey, we provide the frequently used queries for each task (*e.g.,* Table Table 5.8) and the corresponding summarized videos from the TechTube and the original video from YouTube. We then ask the participants to compare the accuracy, preciseness, conciseness and usefulness of each video summary against the original video by answering several questions.

Table Table 5.8 Search Queries Collected from the Developer study

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

Study Findings

All 16 programmers successfully performed the assigned tasks and developed them. Based on the provided screen videos, we find that each task implementation took an average of 26 minutes.

We use a Likert scale of 1 to 10 to capture a participant's responses on the accuracy, preciseness, conciseness and usefulness of a video summary. These criteria are defined as follows:

- **Accuracy:** Measures the extent to which TechTube's search engine retrieves the most relevant videos against the users' natural language search queries.
- **Preciseness:** Measures the extent to which the output of the TechTube (*i.e.,* relevant fragment of the videos) is relevant to the search query.
- **Conciseness:** Measures the extent to which the retrieved relevant fragment of the videos has rich content with respect to their shorter duration time.
- **Usefulness:** Measures the extent to which Techtube's retrieved relevant fragments helps the developers to address their tasks.

Figure Figure 5.6 presents the average and median to which belong participants' responses to each question in the second phase of the study. From Figure Figure 5.6, we realize that, on average, participants find the summarized videos accurate and relevant to the search queries. Furthermore, with a comparison between the original videos and the summarized videos, on average, they discover the video summaries to be precise and concise enough and useful for implementing their programming tasks.

A follow-up survey with the developers reveals that they find the video summaries to be the most useful.

For example, P20 (*i.e.,* participant number 20) commented that: *"TechTube is an amazing tool; I was able to navigate and develop the codes."*

P15 also commented that: *"The summarized videos are handy because of removing the irrelevant information thoroughly for solving the problem."*

The participants also find the usage of video summaries to be an easier and faster method in problem-solving.

For example, P15 commented that *"watching multiple summarized videos discussing a topic is easier and faster to find the solutions rather than using the available resources on the Internet (e.g., forums, YouTube)."*

Besides the mentioned comments, the participants also made several suggestions to extend TechTube for it to become even more useful.

For example, P2 suggested: *"TechTube needs to gather related Stack Overflow posts and code snippets as an enhancement."*

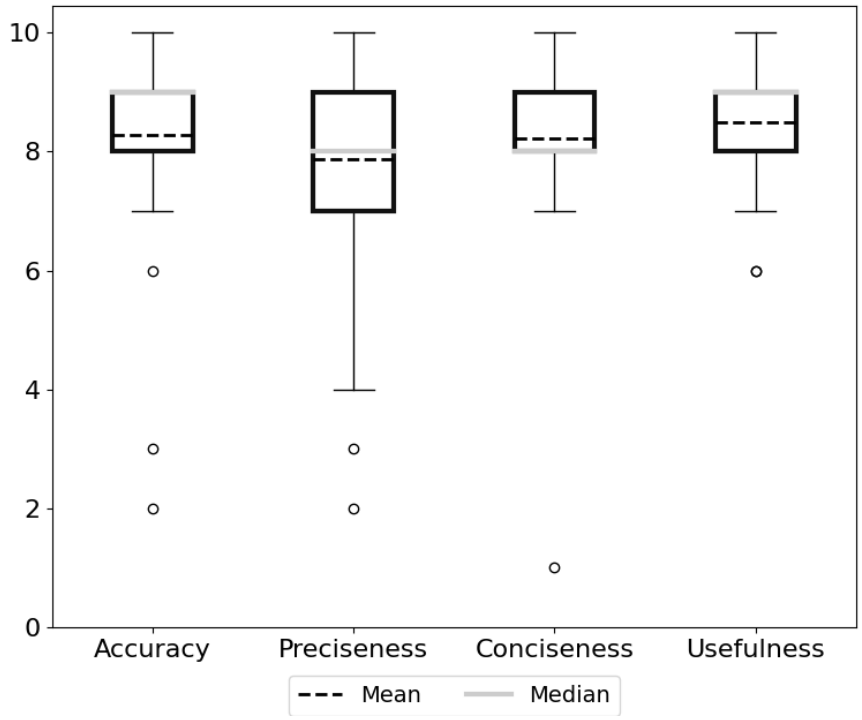

Figure Figure 5.6 Developers' responses on the relevant video summaries generated by TechTube

P13 further clarifies: *"developers need codes rather than the concepts. So TechTube needs to provide codes beside its videos."*

The biggest concern of developers while using TechTube is the lack of the ability to copy and paste the codes.

Besides this, several participants suggested providing an option to select the number of videos to show, to improve the GUI, and to provide more information such as video description and number of views.

In summary, users of TechTube found the video summaries satisfying, precise, and useful for solving their technical tasks.

Summary of RQ_5: The user study involving 16 developers demonstrates that TechTube can assist developers effectively in completing diverse coding and technical tasks. Each participant was able to complete the coding tasks assigned to them using TechTube. By heeding their comments, we can understand that the participants found the relevant video summaries from TechTube useful to complete their tasks.

5.8 Threats to Validity

Internal validity threats relate to experimental errors. In our empirical evaluation of the retrieval of relevant videos and video segments, we compare the results against a ground truth database that we created by taking inputs from three human coders. To mitigate subjective bias, each ground truth was consulted with all three coders. Each of these human coders is a graduate student in Software Engineering. They have the necessary programming background and expertise in Python and Java to analyze the search queries and the videos.

We also mitigated the bias in the user study by asking each participant to complete three programming tasks, where each participant worked remotely without assistance or support from others. In addition, to mitigate the fatigue that may arise during the study, we asked each participant to take sufficient breaks between each task.

External validity threats relate to the generalization of the obtained results. Our approach warrants audio in a video to perform well with the natural language queries.

We use the speaker's silence to split the video (Section 4.1.1). Thus, our approach might not be applicable for online technical videos that either do not contain any audio or contain very little audio. While the current implementation of TechTube uses technical videos from YouTube, it can use technical videos from any online video repository.

Another threat comes from our manual selection of relevant videos for the experiment. Although we carefully identify the relevant videos from a total of 400 videos, our selection might have some subjective bias due to the use of multiple people in tagging the sections of the videos.

We have used the top StackOverflow posts titles as part of our experimental search queries where a number of them might be edited by the authors of the posts to make them more understandable for the responders leading to having more precise results for our experiments.

Reliability threats concern the replicability of our results. We share our experimental data and replication package online [8].

CHAPTER 6 CONCLUSION

6.1 Summary

Developers spend almost 20% of their time finding resources addressing their technical problems through the web. They likely intend to use text-based materials (*e.g.,* question and answer threads in Stack Overflow) over technical video tutorials. One reason for this preference is the easiness of finding appropriate sections of the suggested solutions. Scrolling over the text-based contents and finding the most relevant keywords are less time-consuming rather than skipping the video sections to discover the main solution. Since video tutorials provide audiovisual features, users consider them as one of the vital resources to address their technical tasks such as learning new topics, addressing programming problems, and etc. Thus, identifying the most relevant sections of the video tutorials with less time duration will require the needs of the software developers to find solutions for their technical problems. Accordingly, in this research, we propose a novel approach – TechTube – that identifies the relevant sections of a technical video tutorial to a search query and delivers them as a summarized coherent, video fragment.

TechTube gives developers the capability of using natural language queries to retrieve the most related videos and their relevant sections. To create a precise and concise fragment, TechTube normalizes and reformulates the search query and matches it with technical videos available in an online repository. TechTube uses a silence-based method to better identify the relevant sections of a technical video tutorial.

Reported results of TechTube evaluation against 98 user-generated search queries show that TechTube can retrieve the relevant video tutorials in 93% of the time with a mean reciprocal rank of 0.76 and a mean average of 76%. TechTube can also properly highlight the relevant video segments in a video with a precision of 67% and a recall of 53%. We compared TechTube with a state of the art approach from the literature –CodeTube– and we have presented that TechTube outperforms this approach with 6% higher precision, almost 200% higher recall and over 100% higher F1-score.

We conducted a user study involving 16 developers to evaluate the effectiveness of TechTube during the completion of six development tasks. Reported results show that all the study participants were able to complete the development tasks using TechTube. Regarding the results of a follow-up survey, we have discovered that they also find the summarized videos precise and concise enough to use them in implementing programming tasks.

6.2 Future Work

Our future work will focus on extending TechTube to check for the efficiency of different query retrieval techniques within the TechTube search engine module. We consider using different up to date methods for the natural query reformulation engine of our approach to improve the performance of the tool.

We also will perform a study on how TechTube can complement traditional text-based software documentation resources. This phase will give the ability to use text-based materials besides videos for better addressing their issues (*i.e.,* copy and paste the implemented code as a solution to the users' problems).

Since we did not have any assumptions regarding the number of retrieved videos against a search query, we opted for showing only five of them as the outputs of TechTube. In the future, we plan to access different video platforms (e.g., YouTube, Coursera, Pluralsight, etc) using their available APIs to process and provide several videos (as many as requested by a user) at a time against the users' natural language search queries. By performing such a process, we will improve the scalability of TechTube. Besides, we will improve the architecture of TechTube to accommodate concurrent users' requests with a good performance.

TechTube utilizes a search engine to retrieve the relevant videos. Thus, adapting the search engine to retrieve more appropriate videos according to the users' feedback is a potential future research direction. We assume that the use of Reinforcement Learning (i.e., receiving feedback from users about the content of the videos as a reward and employing it for search engine auto-adaptation) will improve the accuracy of the search engine in retrieving relevant videos.

Videos are one of the most space consuming materials. One way to reduce the cost of processing the videos in terms of disk usage for downloading and storing all the video related files (*e.g.,* videos, audios, speeches, metadata, subtitles and etc) is to perform all the processes in real-time. Thus, we consider building a platform that gives the ability of the real-time process to TechTube approach and uses all the available online technical video tutorials without storing them.

REFERENCES

[1] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code," in *Proc. SIGCHI*, 2009, pp. 1589–1598.

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

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

[4] L. Ponzanelli, G. Bavota, A. Mocci, M. Di Penta, R. Oliveto, M. Hasan, B. Russo, S. Haiduc, and M. Lanza, "Too long; didn't watch! extracting relevant fragments from software development video tutorials," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 261–272. [Online]. Available: https://doi.org/10.1145/2884781.2884824

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

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

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

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

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

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

[11] A. Hauptmann and M. Smith, "Text, speech, and vision for video segmentation: The informedia project," in *Proceedings of AAAI Fall 1995 Symposium on Computational Models for Integrating Language and Vision*, November 1995.

[12] J. S. Boreczky and L. D. Wilcox, "A hidden markov model framework for video segmentation using audio and image features," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, vol. 6, 1998, pp. 3741–3744 vol.6.

[13] Jincheng Huang, Zhu Liu, and Wang Yao, "Integration of audio and visual information for content-based video segmentation," in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, 1998, pp. 526–529 vol.3.

[14] G. Chávez, M. Cord, F. Precioso, S. Philipp-Foliguet, and A. Araújo, "Video segmentation by supervised learning," *Graphics, Patterns and Images, SIBGRAPI Conference on*, vol. 0, pp. 365–372, 10 2006.

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

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

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

[18] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*,

ser. CHI '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1589–1598. [Online]. Available: https://doi.org/10.1145/1518701.1518944

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

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

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

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

[23] M. M. Rahman, "Supporting Source Code Search with Context-Aware and Semantics-Driven Query Reformulation," Ph.D. dissertation, University of Saskatchewan, Canada, 2019.

[24] N. Ksentini, M. Tmar, and F. Gargouri, "The impact of term statistical relationships on rocchio ' s model parameters for pseudo relevance feedback," 2016.

[25] J. Miao, J. X. Huang, and Z. Ye, "Proximity-based rocchio's model for pseudo relevance," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 535–544. [Online]. Available: https://doi.org/10.1145/2348283.2348356

[26] J. Singh and A. Sharan, "Relevance feedback based query expansion model using borda count and semantic similarity approach," *Computational Intelligence and Neuroscience*, vol. 2015, 2015.

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

[28] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11–21, 1972.

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

[30] G. Kumaran and V. R. Carvalho, "Reducing long queries using query quality predictors," in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 564–571. [Online]. Available: https://doi.org/10.1145/1571941.1572038

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

[32] K. C. Youm, J. Ahn, J. Kim, and E. Lee, "Bug localization based on code change histories and bug reports," in *2015 Asia-Pacific Software Engineering Conference (APSEC)*, 2015, pp. 190–197.

[33] S. Zamani, S. P. Lee, R. Shokripour, and J. Anvik, "A noun-based approach to feature location using time-aware term-weighting," *Information and Software Technology*, vol. 56, no. 8, pp. 991–1011, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584914000688

[34] V. Balachandran, "Query by example in large-scale code repositories," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015, pp. 467–476.

[35] R. Lapeña, J. Font, F. Pérez, and C. Cetina, "Improving feature location by transforming the query from natural language into requirements," in *Proceedings of the 20th International Systems and Software Product Line Conference*, ser. SPLC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 362–369. [Online]. Available: https://doi.org/10.1145/2934466.2962732

[36] Z. Lin, Y. Zou, J. Zhao, and B. Xie, "Improving software text retrieval using conceptual knowledge in source code," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2017, pp. 123–134.

[37] S. Wang, D. Lo, and L. Jiang, "Autoquery: automatic construction of dependency queries for code search," *Automated Software Engineering*, vol. 23, pp. 393–425, 2014.

[38] M. Würsch, G. Ghezzi, G. Reif, and H. C. Gall, "Supporting developers with natural language queries," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 165–174. [Online]. Available: https://doi.org/10.1145/1806799.1806827

[39] "Design and evaluation of a multi-recommendation system for local code search," *J. Vis. Lang. Comput.*, vol. 39, no. C, p. 1–9, Apr. 2017. [Online]. Available: https://doi.org/10.1016/j.jvlc.2016.07.002

[40] Meili Lu, X. Sun, S. Wang, D. Lo, and Yucong Duan, "Query expansion via wordnet for effective code search," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2015, pp. 545–549.

[41] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, p. 39–41, Nov. 1995. [Online]. Available: https://doi.org/10.1145/219717.219748

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

[43] S. Wang, D. Lo, and L. Jiang, "Active code search: Incorporating user feedback to improve code search relevance," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 677–682. [Online]. Available: https://doi.org/10.1145/2642937.2642947

[44] S. Haiduc, G. Bavota, A. Marcus, R. Oliveto, A. De Lucia, and T. Menzies, "Automatic query reformulations for text retrieval in software engineering," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 842–851.

[45] E. Hill, L. Pollock, and K. Vijay-Shanker, "Automatically capturing source code context of nl-queries for software maintenance and reuse," in *2009 IEEE 31st International Conference on Software Engineering*, 2009, pp. 232–242.

[46] D. R. Swanson, "Information retrieval as a trial-and-error process," *The Library Quarterly*, vol. 47, pp. 128 – 148, 1977.

[47] R. Mihalcea and P. Tarau, "TextRank: Bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 404–411. [Online]. Available: https://www.aclweb.org/anthology/W04-3252

[48] R. Blanco and C. Lioma, "Graph-based term weighting for information retrieval," *Inf. Retr.*, vol. 15, no. 1, p. 54–92, Feb. 2012. [Online]. Available: https://doi.org/10.1007/s10791-011-9172-x

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

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

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

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

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

[54] S. Haiduc and A. Marcus, "On the effect of the query in ir-based concept location," in *2011 IEEE 19th International Conference on Program Comprehension*, 2011, pp. 234–237.

[55] J. Yang and L. Tan, "Inferring semantically related words from software context," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, 2012, pp. 161–170.

[56] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 3111–3119.

[57] M. M. Rahman and C. K. Roy, "Quickar: Automatic query reformulation for concept location using crowdsourced knowledge," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, pp. 220–225.

[58] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, "From word embeddings to document similarities for improved information retrieval in software engineering," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 2016, pp. 404–415.

[59] Y. Tian, D. Lo, and J. Lawall, "Automated construction of a software-specific word similarity database," in *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, 2014, pp. 44–53.

[60] S. Clinchant and F. Perronnin, "Aggregating continuous word embeddings for information retrieval," in *CVSM@ACL*, 2013.

[61] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones, "Word embedding based generalized language model for information retrieval," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 795–798. [Online]. Available: https://doi.org/10.1145/2766462.2767780

[62] X. Wei and W. B. Croft, "Lda-based document models for ad-hoc retrieval," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 178–185. [Online]. Available: https://doi.org/10.1145/1148170.1148204

[63] N. Rekabsaz, M. Lupu, and A. Hanbury, "Uncertainty in neural network word embedding: Exploration of threshold for similarity," *ArXiv*, vol. abs/1606.06086, 2016.

[64] G. Zuccon, B. Koopman, P. Bruza, and L. Azzopardi, "Integrating and evaluating neural word embeddings in information retrieval," in *Proceedings of the 20th Australasian Document Computing Symposium*, ser. ADCS '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2838931.2838936

[65] A. Berger and J. Lafferty, "Information retrieval as statistical translation," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research*

and Development in Information Retrieval, ser. SIGIR '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 222–229. [Online]. Available: https://doi.org/10.1145/312624.312681

[66] G. Zheng and J. Callan, "Learning to reweight terms with distributed representations," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 575–584. [Online]. Available: https://doi.org/10.1145/2766462.2767700

[67] D. Metzler and W. B. Croft, "A markov random field model for term dependencies," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 472–479. [Online]. Available: https://doi.org/10.1145/1076034.1076115

[68] H. Zamani and W. B. Croft, "Embedding-based query language models," in *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ser. ICTIR '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 147–156. [Online]. Available: https://doi.org/10.1145/2970398.2970405

[69] V. Lavrenko and W. B. Croft, "Relevance based language models," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 120–127. [Online]. Available: https://doi.org/10.1145/383952.383972

[70] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162

[71] M. Almasri, C. Berrut, and J. Chevallet, "A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information," in *ECIR*, 2016.

[72] Y. Zhang, M. M. Rahman, A. Braylan, B. Dang, H. Chang, H. Kim, Q. McNamara, A. Angert, E. Banner, V. Khetan, T. McDonnell, A. T. Nguyen, D. Xu, B. C. Wallace, and M. Lease, "Neural information retrieval: A literature review," *ArXiv*, vol. abs/1611.06792, 2016.

[73] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, vol. 41, no. 6, pp. 391–407, 1990.

[74] E. Nalisnick, B. Mitra, N. Craswell, and R. Caruana, "Improving document ranking with dual word embeddings," in *Proceedings of the 25th International Conference Companion on World Wide Web*, ser. WWW '16 Companion. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, p. 83–84. [Online]. Available: https://doi.org/10.1145/2872518.2889361

[75] I. Vulić and M.-F. Moens, "Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 363–372. [Online]. Available: https://doi.org/10.1145/2766462.2767752

[76] E. Peiszer, T. Lidy, and A. Rauber, "Automatic audio segmentation: Segment boundary and structure detection in popular music," in *Proceedings of the International Workshop on Learning the Semantics of Audio Signals*, vol. 106. LSAS Paris, 2008, pp. 45–59.

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

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

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

[80] T. Liu and J. Guo, "Text similarity computing based on standard deviation," in *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 456–464.

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

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

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

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

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

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

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

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

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