**A PLATFORM FOR IMPROVING SEARCHABILITY AND INTERACTIVITY OF RECORDED LECTURES**

Damsith Karunaratne, Isuru Hettiarachchi, Stefinie Fernando, Sachini Epa

IT16037434, IT15146366, IT16001862, IT16009646

BSc (Hons) in Information Technology

Specializing in Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

August 2019

# Declaration

We declare that this is our own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, we hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute our dissertation, in whole or in part in print, electronic or other medium. We retain the right to use this content in whole or part in future works (such as articles or books).

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Signature |
| Karunaratne D. C. | IT 16037434 |  |
| Hettiarachchi H. A. I. S. | IT 15146366 |  |
| Fernando S. S. M. S. | IT 16001862 |  |
| Epa S. S. | IT 16009646 |  |

The above candidate has carried out research for the bachelor’s degree Dissertation under my supervision.

Supervisor: Dr. Nuwan Kodagoda :

Co-Supervisor Ms. Kushnara Suriyawansa :

Date:

# Acknowledgement

The work described in this research paper was carried out as part of the final year research project for the subject Comprehensive Design Analysis Project. The completed final project is the result of combining all the hard work of the group members and the encouragement, support and guidance given by many others. Therefore, it is our duty to express our gratitude to all who supported us in this endeavor. We would like to thank Dr. Nuwan Kodagooda and Ms. Kushnara Suriyawansa our supervisor and co-supervisor, for the invaluable advice and guidance they provided. We are also extremely grateful to Mr. Jayantha Amararachchi, Senior Lecturer/ Head-SLIIT Centre for Research who gave and confirmed the permission to carry out this research. We also wish to thank all our colleagues and friends for their support. Finally, we would like to thank all others whose names are not listed particularly but have given their support in many ways and encouraged us to make this project a success.

# Abstract

E-Learning has become commonplace and many leading Universities provide the facility to view pre-recorded lectures online. This approach gives learners the ability to follow lectures without time or location constraints and consume the lectures at their own pace. Despite their advantages, recorded lectures tend to be lengthy and tedious to watch. They also prove cumbersome when specific information needs to be extracted from them. Another drawback is that the lecture videos fail to show the connection between the lecture and its supporting material such as lecture slides and questionnaires.

Several platforms exist where videos can be edited to make them more interactive, however this is a time-consuming process. We propose a system which will automatically improve the interactivity and accessibility of recorded lectures in a few clicks. The system will take in raw lecture videos along with supporting material such as lecture slides and code samples. It will then carry out noise removal and optimizing on the raw video footage before matching the slides and code samples to occurrences in the video. Some novel features we plan on introducing are automatic generation and suggestion of questions based on the content and automatic video segmentation according to topics. The main objective of the system is to create a web platform which can add interactivity and accessibility to course material thereby improving learner engagement.

# Table of Contents

[Declaration i](#_Toc16546113)

[Acknowledgement ii](#_Toc16546114)

[Abstract iii](#_Toc16546115)

[Table of Contents iv](#_Toc16546116)

[List of figures vii](#_Toc16546117)

[List of tables viii](#_Toc16546118)

[List of abbreviations ix](#_Toc16546119)

[1 Introduction 1](#_Toc16546120)

[1.1 Background 1](#_Toc16546121)

[1.2 Literature Survey 2](#_Toc16546122)

[1.3 Research gap 7](#_Toc16546123)

[1.4 Research problem 8](#_Toc16546124)

[1.5 Research objectives 8](#_Toc16546125)

[2 Methodology 10](#_Toc16546126)

[2.1 Methodology 11](#_Toc16546127)

[2.1.1 Code Matching 11](#_Toc16546128)

[2.1.1.1 Frame Extraction 11](#_Toc16546129)

[2.1.1.2 Building the Image Classifier to filter frames 12](#_Toc16546130)

[2.1.1.3 Code Matching algorithm 15](#_Toc16546131)

[2.1.2 Content Based Video Segmentation into Topic Units 17](#_Toc16546132)

[2.1.2.1 Transcribing Videos 17](#_Toc16546133)

[2.1.1.1 Topic Modelling 20](#_Toc16546134)

[2.1.2.2 Text Segmentation 21](#_Toc16546135)

[2.1.2.3 Reviewing and editing suggested topics 23](#_Toc16546136)

[2.1.3 Automatic Question Generation 24](#_Toc16546137)

[2.1.3.1 Extracting Input Texts 25](#_Toc16546138)

[2.1.3.2 Question Generation 26](#_Toc16546139)

[2.1.3.3 Question Reviewing and Presenting to the User 30](#_Toc16546140)

[2.1.4 Slide Matching 30](#_Toc16546141)

[2.1.4.1 Methodology 31](#_Toc16546142)

[2.1.4.2 Algorithm 32](#_Toc16546143)

[2.1.4.3 Audio Denoising 35](#_Toc16546144)

[2.2 Commercialization Aspects of the Product 36](#_Toc16546145)

[2.3 Testing and implementation 36](#_Toc16546146)

[3 Results and discussion 37](#_Toc16546147)

[3.1 Results 37](#_Toc16546148)

[3.1.1 Results for the Code Matching Component 37](#_Toc16546149)

[3.1.2 Results for Automatic Question Generation Component 39](#_Toc16546150)

[3.1.3 Topic Modelling and Segmentation 40](#_Toc16546151)

[3.1.3. Results for Slide Matching Component 43](#_Toc16546152)

[3.1.3.1.Measuring performance 43](#_Toc16546153)

[3.1.3.2. Measuring accuracy 44](#_Toc16546154)

[3.2 Research Findings 45](#_Toc16546155)

[3.3 Discussion 45](#_Toc16546156)

[4 Conclusion 48](#_Toc16546157)

[5 References 49](#_Toc16546158)

[6 Appendices 56](#_Toc16546159)

# List of figures

[Figure 2.1: High-level system diagram 10](https://mysliit.sharepoint.com/sites/cdap-2019-lecturevideos/Shared%20Documents/Documents/19-087_Thesis_Draft.docx#_Toc16545206)

[Figure 2.2 - High level architecture of code matching system 11](#_Toc16545207)

[Figure 2.3 - Extracting frames 12](#_Toc16545208)

[Figure 2.4- VGG16 model architecture 14](#_Toc16545209)

[Figure 2.5- Resnet model architecture 15](#_Toc16545210)

[Figure 2.6 - InceptionV3 architecture 15](#_Toc16545211)

[Figure 2.7- Pseudocode of code-matching algorithm 16](#_Toc16545212)

[Figure 2.8: Sample timestamp from AssemblyAI 19](#_Toc16545213)

[Figure 2.9: Plate notation for LDA with Dirichlet-distributed topic-word distributions 21](https://mysliit.sharepoint.com/sites/cdap-2019-lecturevideos/Shared%20Documents/Documents/19-087_Thesis_Draft.docx#_Toc16545214)

[Figure 2.10: Sample output from topic tiling algorithm 22](https://mysliit.sharepoint.com/sites/cdap-2019-lecturevideos/Shared%20Documents/Documents/19-087_Thesis_Draft.docx#_Toc16545215)

[Figure 2.11: High Level view of overall Topic modelling and segmentation process 23](https://mysliit.sharepoint.com/sites/cdap-2019-lecturevideos/Shared%20Documents/Documents/19-087_Thesis_Draft.docx#_Toc16545216)

[Figure 2.12: VTutor video review UI 24](#_Toc16545217)

[Figure 2.13: High-level diagram of automatic question generation process 24](#_Toc16545218)

[Figure 2.14: Question creation process by transforming sentences 28](https://mysliit.sharepoint.com/sites/cdap-2019-lecturevideos/Shared%20Documents/Documents/19-087_Thesis_Draft.docx#_Toc16545219)

[Figure 2.15: User interface question review 30](#_Toc16545220)

[Figure 2.16: Slide Matching Process Overview 31](#_Toc16545221)

[Figure 2.17: User Interface for Slide Matching 35](#_Toc16545222)

[Figure 8- InceptionV3 comparison 38](#_Toc16545223)

[Figure 9- Resnet50 comparison 38](#_Toc16545224)

[Figure 10 - VGG16 comparison 39](#_Toc16545225)

# List of tables

[Table 1.1: Comparison of existing products 7](#_Toc16545196)

[Table 1- Comparison of candidate models 37](#_Toc16545197)

[Table 3.1: Qualitative test results for question items generated in automatic question generation component 40](#_Toc16545198)

[Table 3.2 Word confidence level from AssemblyAI 40](#_Toc16545199)

[Table 3.3 List of lowest Pk values for the Choi data set for different algorithms. 41](#_Toc16545200)

[Table 3.4 Sample 1 text segmentation results 42](#_Toc16545201)

[Table 3.5 Sample 2 text segmentation results 42](#_Toc16545202)

[Table 3.6 Sample 3 text segmentation results 43](#_Toc16545203)

[Table 8: Performance comparison - OOC Lecture 5 43](#_Toc16545204)

[Table 9:Performance comparison - OOC Lecture 6 44](#_Toc16545205)

# List of abbreviations

API Application Programming Interface  
RAM Random Access Memory  
MOOC Massive Open Online Course  
NLP Natural Language Processing  
SaaS Software as a Service  
URL Unified Resource Location  
AWS Amazon Web Services  
GHz Gigahertz  
BSc Bachelor of Science

# Introduction

## Background

Today, e-learning has become an essential component of higher education for both teachers and students. According to a study on the effectiveness of e-learning on education, it was found that students nowadays are more satisfied with web enhanced learning when compared to a traditional classroom environment [2]. Therefore, it is common to see universities and higher education institutes adopting some form of e-learning to assist their students. Many institutes use their own customized version of a Learning Management System (LMS) to provide online course material.

Online education is beneficial for both students and teachers in many ways. For instance, as lecture content is always available online, the possibility of missing a lecture is low and teachers can ensure that students have access to course material irrespective of time and location [2]. Recorded lecture videos also enable students with different styles of learning and different levels of understanding to obtain a better grasp of the subject. For example, those who are familiar with the work can skip ahead to a section of interest while those that need more time to understand the concepts can pause and rewind to digest the lecture at their own pace [3, 4].

Usually, many LMSs enable lecturers to upload course material such as tutorials, lab sheets, lecture slides and recorded lecture videos. Whilst videos are more effective because they address both visual and auditory aspects of teaching [5], many students find it tedious to watch recorded lecture videos because of its duration, which normally lasts around 1 - 3 hours and its lack of interconnectivity and relevance to other course material [6]. Although there are many platforms for creating and editing videos which allow users to create interactive course material, the methods employed by them consume valuable time as they require the lecturer to manually edit the video.

At the time of writing this proposal, to the best of our knowledge, there is no system which automatically identifies the relationship between different types of course material and enables the creation of interactive courses in a few steps. Hence, the focus of our research project is on improving two aspects of course material: accessibility and interactivity. The goal is to develop an intelligent system capable of improving the interactivity and learner engagement of course material in just a few clicks.

## Literature Survey

**Matching each line of code in a sample code file to its occurrence in a live-coding video**

Live coding lectures, often seen in the field of Information Technology, usually feature a screen capture in which the instructor types code into a text editor or Integrated Development Environment (IDE) while narrating. A study conducted by Marc. J. Rubin, on the effectiveness of live-coding lectures, found that students exposed to live-coding lectures performed better when tackling large programming assignments [7].

Although lectures of this type have their merits, students often need to revisit a specific point in a lecture and to do this they must search through the video until the relevant section is found. This has been identified as a drawback and many researchers have tried to address this problem with varying degrees of success. In a research conducted in 2011, Kambathula and Iyer suggested a system which would enable automatic tagging of lecture videos to enable easy identification of the different sections. It achieves this by first performing text analysis on the audio extracted from the video and then creating a database of tags from the resulting transcript [8]. Their system can highlight portions in each video in response to user queries, allowing the user to navigate to an exact location in the video. Luca Ponzanelli *et al* in 2016 introduced CodeTube [9], a similar search engine which when given a query, returns self-contained fragments of the corresponding lecture videos. Their system can identify Java code in video frames by applying image processing techniques such as Optical Character Recognition (OCR) and shape detection as well as text analysis methods such as island parsing on subsections of each frame to generate a Heterogenous Abstract Syntax Tree (H-AST) which is used to identify coding constructs [10]. In a research conducted in 2018, a deep-learning approach which leverages Convolutional Neural Networks (CNNs) to classify the presence or absence of Java code in video frames is proposed [11]. Their system is able to achieve an average accuracy of 98% for this binary classification task using a trained VGG16 [12] Neural Network and represents a more scalable solution to identifying code in videos. However it is limited to identifying code in the Java language and cannot be successfully applied in a system which would analyze videos in many programming languages. Research on algorithms such as ResNet50 [13] and InceptionV3 [14] prove they are good candidates for this purpose.

As illustrated above, much research has been conducted on identifying source code in video and image files. Although the topic area (matching each line of code in a sample code file to its occurrence in the lecture video) has not been widely researched and would most likely have an algorithmic solution, research that has already been conducted on source code mining and text detection can be used as a basis to create a system which identifies relevant portions of a live-coding video which correspond to the source code file.

**Content-based Lecture Video Segmentation into topic units**

Segmentation of long lecture videos into cohesive topic units is highly beneficial since it makes the search for information easier, minimizes learning time and improves the overall learning experience. Research has been carried out in relation to segmenting videos based mainly on three areas - visual content, audio and text (transcripts). However, limited research has been conducted in relation to the specific domain of segmenting lecture videos into topic units.

The most commonly used mechanism for segmenting videos is based on scene changes. A research conducted in 2000 used color histogram distance computation between successive images to detect scene changes [15]. Another research done by Zhang and Smoliar, proposed a system for detecting progressive transitions based on both motion and statistical analysis [16]. However, this mechanism of segmentation based on scene changes is not applicable to lecture videos, as they have very few scene changes and even these scene changes do not match with topic transitions [5].

Another feature considered by researchers for lecture video segmentation is slide matching. A research done in 2013, focused on segmenting lecture videos into topics by analyzing its supplementary synchronized slides. They used OCR to extract content from lecture slides and identify different subtopics according to their logical relevance. Slides were then synchronized with the video stream to identify different topic changes. As stated in their paper [17] the mechanism was approximately 90% accurate, except that it always assumes that the slides are synchronized with the video streams, which, in practice might not always be the case. Furthermore, since their method was solely based on matching slide content with the video, its accuracy is limited only for certain lecture video types.

Moreover, segmentation methods utilizing transcribed text or closed captions have also been researched on. The main motivational factor for work in this area was the Topic Detection and Tracking (TDT) initiative conducted in 1998 [15]. TDT is defined as the task of segmenting transcribed speech into topically cohesive stories. Their algorithm is trained mainly using broadcast and news domain data sets where, formal presentation format and cue phrases are used to improve segmentation accuracy. Unlike in the broadcast domain, speeches in lecture videos are often unscripted and spontaneous. Furthermore, a large training dataset is used for many methods in TDT, which is not available for lecture videos [5, 17, 18].

Inspired by the work on TDT, in 2004 Michael Chau, and his team members, conducted another research to identify topic changes based on multiple linguistic features like noun phrases, topic noun phrases, verb classes, word stems, combined features, cue phrases, and pronouns [5]. Using automatic speech recognition software, they retrieved timestamps that synchronize with the video stream and then the results from transcribed text segmentation is then mapped back to video segmentation. However, they only developed a set of algorithms each considering one specific feature out of the list of segmentation features mentioned above. They then compared the results from these algorithms to identify the most salient feature for lecture video segmentation. Work in this area can be considered as a potentially successful solution and can be improved further as audio and the transcribed text extracted from lecture videos provide rich content information for topic change detection [5].

**Automatic question generation**

The main goal of any lecture is that the learner should achieve knowledge on a specific subject or area. This specific knowledge gained from a lecture is known as its learning outcome. One of the primary and most effective ways to evaluate whether the learner has achieved these learning outcomes is quizzes. However, formulation of questions for quizzes can be a time-consuming task.

Research has been carried out on ways to automatically generate questions. Shah et al suggest a method of generating Multiple Choice Questions (MCQs) for an input text passage with the aid of a one-time trained knowledge base developed using Wikipedia articles [19]. During the execution of the system words in sentences are mapped to a predefined dictionary and an Inverse Document Frequency score is used to choose the word that serves as a blank. Distractors are generated by the paradigmatic discovery on a self-made corpus and dictionary.

TEDQuiz [20] is a system that generates MCQs for TED Talks video clips using a graph-based algorithm. The system generates two types of questions. The first type is gist-content questions which ask about the overall theme of the content that is generated by identifying the most important sentence using LexRank [21] and creating distractors by less important sentences. The second type is detailed questions which use Heilman and Smith’s work [22, 23] to create question stem and selecting words for distractors using WordNet and similar corpus.

A similar system which analyzes a text transcript of a video lecture to suggest self-assessment items at runtime is seen in [24]. The process is carried out by identifying discourse boundaries from the lecture and retrieving Wikipedia text segments related to identified discourse boundaries for further well-formed and formal discourse to generate question items: MCQs generated using Heilman’s work [25] and distractors generated on an ontology-based strategy using Wikipedia category taxonomy as a replacement for the ontology.

Apart from this, automatic question generation was carried out using ontology-based strategies. SeMCQ [26] is a Protégé plugin created for automatic ontology-driven multiple question test generation. OntoQue [27] is an automatic question generation engine based on domain ontologies which can generate MCQ, true/false questions and fill-in-the-blank questions. Papasalouros et al. suggest an approach of generating MCQs based on domain-specific ontologies that use simple natural language generation techniques [28].

**Synchronization of slides with the lecture video and denoising**

Most of the time, lecturers upload PowerPoint presentations along with the lecture video for reference. For easy access, it is highly beneficial to have the ability to point out the occurrences where the slide is discussed in the lecture video, when a particular slide is selected rather than manually going through the entire lecture video to find out the occurrences in which the particular slide is discussed. In addition, removing unnecessary noises like the noise of breathing and time intervals with no audio would optimize the time taken for the learning process.

Several research projects have been carried out related to synchronization of lecture slides with the video. Among them, a research has been done by the Hongkong university and science technology and City university of Hongkong in which a system using OCR is proposed [29]. In the system, geometry-based approach for text detection which works well with images with less noise and Super Resolution Reconstruction approach is used to enhance the visual quality of the video texts to make the video images appropriate for commercial OCR systems.

Zentation [30] is an online tool that can be used to synchronize the slides with the video. But it is not a perfect tool when it comes to synchronization since most of the time this works well with presentations that have a smaller number of slides. Therefore, this kind of tool cannot be used for matching lecture videos with the slides as most of the times lecture slide decks contain a considerable number of slides.

When it comes to denoising audio enhancement studies have been carried out reduce the background noise and to enhance intelligibility [31, 32, 33, 34, 35]. Removal of high frequency noise for speech enhancement with frequency response masking (FRM) has also been implemented. A FIR filter has been designed to have impulse responses associated with various cut off frequencies to minimize the error when comparing the original speech signal and the filtered speech signal [36].

Audacity [37] is a tool that can be used to remove background noise of an audio and it can be used to enhance the audio quality also. It can be used to remove regular noises like static, hum, hiss and other constant background noises. In addition, this tool enables the user to change the pitch without any changes in the tempo. But Audacity is not capable of removing irregular noises like traffic sounds and sounds of an audience which can occur in lecture videos. As a result, tools like Audacity cannot be used to denoise lecture videos in e-learning platforms as it contains irregular noises rather than constant noises.

## Research gap

Even though there are several products already available with similar objectives, they mostly focus on the use of manual processes that involve human intervention to make the content more interactive and increase the searchability. Our proposed solution aims to reduce the amount of human interaction needed for this process by introducing a platform which will analyze and augment the content automatically. Table 1.1 is a comparison of the proposed system with existing systems in the market.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Features | LearnWorlds | Echo360 | TechSmith Relay | Our Solution |
| Matching lines in code samples to occurrence in recorded lectures | ✗ | ✗ | ✗ | ✔ |
| Automated segmentation of lecture video into topic units | ✗ | ✗ | ✗ | ✔ |
| Matching slides with the lecture video | ✗ | ✔ | ✗ | ✔ |
| Automated noise removal from the video | ✗ | ✗ | ✗ | ✔ |
| Automatic question generation | ✗ | ✗ | ✗ | ✔ |

Table 1.1: Comparison of existing products

## Research problem

Nowadays, video lectures have become increasingly popular and many education institutes use Learning Management Systems that support video content. Whilst video lectures have benefits such as giving learners remote access to lectures, there are a few drawbacks such as poor searchability through the video and less interaction with the learner.

To overcome these drawbacks many lecture platforms have introduced tools such as web-based video editors that allow lecturers to add captions, divide the video into discussed topics, link lecture slides and embed questions into the video. However, these tools require human intervention which is time-consuming. When considering the domain of computer science, none of the available platforms provide a tool to map programming language code segments from source code files discussed in a lecture with their occurrences in the video.

Because there is a need for video lectures to be more interactive and searchable, and the fact that enhancing them in such a way requires a significant manual workload and time investment, a platform which will analyze and augment the content automatically will prove to be useful.

## Research objectives

The proposed system is a research study to improve the method of delivery for video lectures and increase the engagement of the learner. The main objective of the research is to develop an automated platform which provides a quick and efficient way for lecture creators to deliver video lectures which are more interactive and have increased searchability. The other objectives of this research are as follows,

* Create an algorithm to automatically identify the position in the lecture video where each line of code in a given related code sample is discussed so that the learner can reach that position on the video by using the lines of code in the code sample as an index.
* Increase the efficiency of the process of code identification by classifying frames which contain code and those that do not, thereby reducing the resources wasted on unnecessary frames.
* Create an algorithm to automatically identify the position of time in the lecture video where a given slide is discussed so that the users get the privilege to traverse through the lecture video using the electronic slides as an index.
* Find a methodology that is able to remove the noise from the raw lecture video to enhance the learning process.
* Increase the interactivity of the lecture by generating question items related to lecture and embed in the video.
* Develop and train a topic model which can effectively identify topics from video transcripts.
* Automatically identify main topic transitions of a given lecture video based on the topic model and segment the video according to the identified positions.

# Methodology

Figure 2.1A screenshot of a cell phone

Description automatically generated shows a high-level system diagram. The system is comprised of cloud based Node.js[[1]](#footnote-2) microservices which form the backend and an AngularJS[[2]](#footnote-3) web application as the frontend. Upon uploading a lecture video with the appropriate lecture material such as slides and source code files, the system first generates a transcript and extracts frames from the video. The frames and transcript are then fed into the Topic segmentation, Question Generation, Code Matching and Slide Matching modules which generate metadata related to the video. The metadata consists of the timestamps corresponding to the matched slides, matched lines of code, generated questions and topic breaks in the video. The generated metadata is stored as a document in a NoSQL MongoDB database while the uploaded video is stored in an Amazon S3 bucket[[3]](#footnote-4). Finally, the frontend web application will access this data using an Application Programing Interface (API) to provide an enhanced lecture viewing experience to the end users

Figure ‑: High-level system diagram

## Methodology

### Code Matching

The code matching component performs the task of identifying the timestamp in the video where a particular line of code is discussed. This task can be broken down into the following steps,

1. Splitting the video into a set of frames which preserves the ordering of each frame in sequence
2. Filtering frames which contain code.
3. Obtaining a textual representation of the content in each frame
4. Iterating through each line in the source code file to find the earliest frame in the frame sequence which contains the line of code.

Figure 2.2 shows a high-level architecture of the code matching component.

A picture containing device

Description automatically generated

Figure 2‑2 - High level architecture of code matching system

#### Frame Extraction

As a prerequisite to step 1, the video is uploaded to Amazon S3. Once the uploading process is complete, we utilize the Amazon Simple Notification Service (SNS)[[4]](#footnote-5) which is a highly available and fully managed pub/sub messaging service to notify the backend server that the video has been uploaded. The notification contains the Unique resource identifier (URI) of the uploaded video which is then used as an input for the ffmpeg[[5]](#footnote-6) library to extract frames. Figure 2.3 shows the javascript code to spawn a child process which calls the ffmpeg library to perform frame extraction. The video is sampled at 0.5 frames per second and each frame is named in sequence. This naming order is required to determine the timestamp of each frame.

A picture containing object

Description automatically generated

Figure 2‑3 - Extracting frames

#### Building the Image Classifier to filter frames

To match a code sample to its occurrence in the lecture video, text detection must be carried out on each video frame to extract a textual representation of it. Existing research in the fields of OCR and Text detection suggests that it is more efficient to use a Machine Learning model to detect candidate frames before running the OCR algorithm [9, 11] instead of wasting CPU cycles on frames which do not contain code. Based on this research we propose to train a machine learning algorithm to detect such frames. Considering the time constraints of the project we will apply Transfer Learning techniques [38, 39] to repurpose an established machine learning model for this classification. The following sub sections describe the steps carried out to create this image classifier.

1. **Creating the Dataset**

The initial step of any supervised classification problem in machine learning is to collect a suitable dataset. Since at the time of writing this report there was no publicly available dataset with images that fall into the classes of *code* and *not-code,* the data had to be generated manually. Generating images was trivial given the abundance of lecture material provided by our supervisor. However manually labeling each image proved to be difficult given the time constraints of the project. Therefore a python script was created to scrape the results of a google image search for each relevant class, download them to a file and ensure that they are valid. The next step was to remove visually similar images by resampling using a Lanczos filter [40] and extracting an average pixel level for each band in the image to create a hash representation of each image. The hashes were then compared to discard similar images. The resulting images were inspected manually to remove any images that were irrelevant to each class. As a result a dataset of 450 images of code and 450 images of other material such as slides which did not contain code was created in a relatively short time period. Furthermore, the Keras pre-processing package [41] was leveraged to perform random transformations and normalization on an image to generate multiple new images, thus increasing the amount of effective training and test data available.

1. **Reusing a pre-trained model with transfer learning**

Most deep learning models which solve complex problems need a vast amount of labeled data which, considering the time and effort required, can prove difficult to collect. Furthermore, training a deep learning model on a large dataset can take days or even weeks. However the application of knowledge gained from pre-trained models to solve related problems also known as *transfer learning*, can drastically reduce the effort needed to train a new model.

Supervised Deep learning models extract different features at different layers. Yosinski et al. In their paper [42], discuss how the lower layers of a neural network act as abstract feature extractors to detect features like edges and curves, while the dense layers towards the end identify features which are more specific to the task it was trained for. Therefore by freezing the weights of a robust pre-trained network and removing the last, fully connected layer, it can act as a feature extractor for similar classification tasks.

Many high-performing models have been developed for image classification for the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Most of the models developed for this challenge are available for free through various deep learning APIs including Keras. Transfer learning was carried out on the VGG16 [12], ResNet50 [13] and InceptionV3 [14], models which are typically used for most image classification experiments. Each model was compared based on its accuracy and validation loss to determine which model is suitable to use as a classifier.

1. **Comparison of architectures**

Several candidate deep learning architectures were analyzed to determine the best fit to classify images as containing code and not containing code. Each proposed model is introduced in this section.

VGG16 is a network which is characterized by its simplicity. The *16* stands for the number of weight layers in the network. Figure 2.4 taken from Simonyan and Zisserman’s paper [12] , depicts the model architecture.

A screenshot of a cell phone

Description automatically generated

Figure 2‑4- VGG16 model architecture

There are two major drawbacks when considering this model which are, the time taken to train the network and the size of the model. VGG16 is over 574MB large.

Resnet50, introduced in the paper “Deep Residual Learning for image recognition” by He et al [13], has a deeper architecture than VGG16. However at 102MB the model is substantially smaller in size due to the usage of global average pooling as opposed to fully-connected layers. Figure 2.5 depicts the layer arrangement of the Resnet architecture.

A screenshot of a cell phone

Description automatically generated

Figure 2‑5- Resnet model architecture

Szegedy et al. introduced the Inception architecture in their 2014 paper [14]. Figure 2.6 depicts a summary of the architecture. Out of all three models discussed, this model has the smallest size at just 96MB.

A screenshot of a cell phone

Description automatically generated

Figure 2‑6 - InceptionV3 architecture

#### 2.1.1.3 Code Matching algorithm

The deep learning model will identify the frames which contain code, thereby forming a set of images ***I****.* For each image ***i* ∈ *I*** the text contained within is extracted using the Optical Character Recognition capabilities of TESSERACT-OCR[[6]](#footnote-7) and stored in a set of text files ***T*** which preserves the file naming scheme of the extracted frames such that the chronological ordering of the frames is not disturbed. For example; *frame-0000.txt* refers to the first frame in the sequence.

The main intuition behind the algorithm is that the code will be discussed at the earliest time it occurs within the video. And therefore each frame should be analyzed from the end of the sequence to the beginning while looking for the best match. Figure 2.7 describes the proposed algorithm which determines the frame which is most suitable to use as the index. In the diagram, *T* refers to the set of text files extracted from the frames and *C* is the source code file which is to be matched.

A screenshot of a cell phone

Description automatically generated

Figure 2‑7- Pseudocode of code-matching algorithm

In the algorithm mentioned above, *gestalt ()* refers to a function based on Ratcliff and Obershelp’s “gestalt pattern matching” algorithm [43]. The algorithm works by first finding the longest common substring (LCS) from the two strings. Then it splits the strings into two parts, one to the left and another to the right of the common substring. Next the process is repeated, first for the left parts of both strings, and then the right parts. The process of finding the LCS is repeated recursively until the size of any split is less than a predefined value. Finally the similarity score is calculated using the following formula. takes a value between 0 and 1 where 1 means that the two strings match completely and 0 means that not even one common letter was found.

= number of characters in both strings

The execution time of the above algorithm is andin the best and worst case respectively. A variation of the algorithm which returns an upper on the value is implemented in the python *difflib* library [44]. It performs this by using the intersect of all common symbols in each string instead of recursively matching each sequence.

This reduces the execution time to in the worst case andin the best case at the marginal sacrifice of accuracy.

shows an example output of the code matching algorithm. Here, each line is provided with its corresponding timestamp location in the video.

**A close up of a map

Description automatically generated**

Figure 2‑8 - Example output of code matching algorithm

### Content Based Video Segmentation into Topic Units

#### Transcribing Videos

A key factor which affects the overall accuracy of topic modelling is how accurately the videos are transcribed. For transcribing videos, a third-party API called AssemblyAI[[7]](#footnote-8) is used. With AssemblyAI a time stamped transcript can be extracted for a given video URL, which contains the time (its occurrence in the video) in milliseconds for each and every word in the video. A sample time stamped transcript generated by AssemblyAI API is shown in Figure 2.2. This time stamped transcript is later used to identify the segmentations point in the video, in other words times where the video needs to be segmented.

The transcript of the video needs to be extracted automatically whenever a new video is uploaded to cloud storage through the web app. To trigger this flow following cloud architecture is used when deploying the web app.

**{ "status": "completed",**

**"acoustic\_model": "assemblyai\_default",**

**"text": "Look at collection classes specifically, we look at an introduction to a list and also what action classes are really. So basically you have a framework, you have a framework of classes in Java which allows you to do pretty sophisticated stuff.",**

**"format\_text": true,**

**"punctuate": true,**

**"audio\_url":"https://cdap-bucket.s3.ap-south-1.amazonaws.com/2019-OOP-Revision-ArrayList-1-MP4.mp4",**

**"words": [**

**{ "text": "Look",**

**"confidence": 0.73,**

**"end": 140,**

**"start": 0**

**},**

**{ "text": "at",**

**"confidence": 0.86,**

**"end": 240,**

**"start": 100**

**},**

**{ "text": "collection",**

**"confidence": 0.87,**

**"end": 640,**

**"start": 220**

**},**

**],**

**"language\_model": "computer-science-model-2",**

**"id": "898o9jwx-01a9-4fc2-b7c9-c5595540adf3",**

**"confidence": 0.894341138659315,**

**"utterances": null,**

**"audio\_duration": 2341.10548752834,**

**"webhook\_status\_code": 200,**

**"webhook\_url": "http://52.66.30.76:5000/vtutor-transcriptions-api/v1/get-transcript",**

**"dual\_channel": null }**

Figure 2‑9: Sample timestamp from AssemblyAI

#### Topic Modelling

Our approach for topic modelling is based on a state of art algorithm known as Latent Dirichlet allocation (LDA). LDA is a form of unsupervised learning and is one of the most popular generative, probabilistic text modelling techniques in machine learning. LDA works with the assumption that each document was generated by picking a set of topics and then for each topic picking a set of words [45]. Input to LDA is a collection of documents, which in our case is the video transcripts.

LDA considers each document is in the form of a ‘bag of words’ [45]. Therefore, in order to transform raw transcripts into a bag of words structure, it need to undergo data pre-processing. Standard data pre-processing for LDA consists of the following steps:

* *Tokenization* - Split text into words, lowercase the words and remove punctuation
* *Remove stopwords* - stopwords are most commonly used words in a language, for an example word ‘the’ in English language. Stopwords are usually not relevant to any topics hence, removing these from the corpus supports LDA to extract topics more accurately.
* *Lemmatization* - third person words are changed to first person, and verbs in different tenses are converted to present tense.
* *Stemming* - words are reduced to their root form.

Once preprocessing is done the corpus is subjected to LDA. The generative process of LDA is as follows:

For each document w in a corpus D:

1. Choose N Poisson (ξ).

2. Choose Θ Dir (α).

3. For each of the N words wn:

a. Choose a topic zn Multinomial (Θ).

Choose a word wn from 𝑝(𝑤𝑛 |𝑧𝑛, 𝛽), a multinomial probability conditioned on the topic zn. [46]

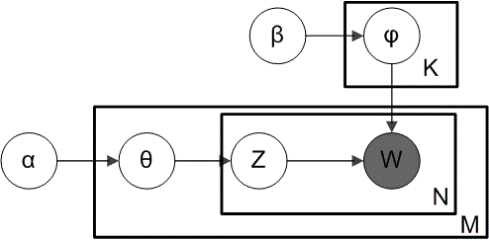


Figure ‑: Plate notation for LDA with Dirichlet-distributed topic-word distributions

Source: <https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation>

α - per-document topic distributions,

β - per-topic word distribution,

θ - topic distribution for document m,

φ - word distribution for topic k,

z - topic for the n-th word in document m

w - specific word [47]

#### Text Segmentation

Text segmentation is achieved through a LDA based segmentation algorithm known as topictiling. Topic Tiling was first introduced by Riedl and Biemann in 2012 and was inspired by the text tiling algorithm developed by Mari Hearst. Unlike in text tiling which segments text based on words, topic tiling algorithm is based on topic IDs assigned by Bayesian Inference method of LDA. When compared with other segmentation methods based on LDA, topic toiling is computationally less expensive as it performs segmentation in linear time. [48]

In topic tiling, initially the document is split into units based on sentence boundaries. Each sentence is represented by a N dimensional vector where N is the number of topics represented in the topic model. Then the coherence between each sentence is computed using cosine similarity between two adjacent sentences [49]. The final outcome of the topic tiling algorithm will be an XML file with text segments as shown in Figure 2.4.

Figure ‑: Sample output from topic tiling algorithm

**<documents>**

**<document>**

**<documentName>input.txt</documentName>**

**<segments>**

**<segment>**

**<depthScore>0.0</depthScore>**

**<text>………………..</text>**

**</segment>**

**<segment>**

**<depthScore>0.012</depthScore>**

**<text>………………..</text>**

**</segment>**

**</segemnts>**

**</document>**

**</documents>**

This XML file is then read by another algorithm and matched with a time stamped transcript of the video to extract the timestamps where the segmentations occur. Final outcome of the topic modelling component will be a set timestamps along with set of key words suggested from the topic model.

The overall topic modelling and text segmentation process is shown in Figure 2.5.

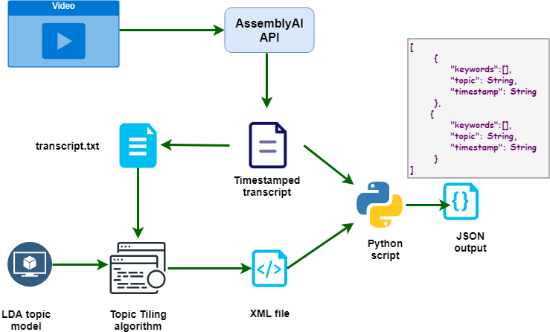


Figure ‑: High Level view of overall Topic modelling and segmentation process

#### Reviewing and editing suggested topics

The overall topic modelling and video segmentation process can only suggest a set key words for each topic and the timestamps where possible topic transitions might occur. It cannot predict the exact topic for a given segment, because topic is something vague and may differ according to an individual’s perspective. Therefore, once a lecture video is processed, it should be reviewed by the lecturer before publishing it for students. Vtutor web app consists of a video review UI Figure 2.12 where the lecturers can review and edit the topics and segmentation points suggested by the system, before publishing it.

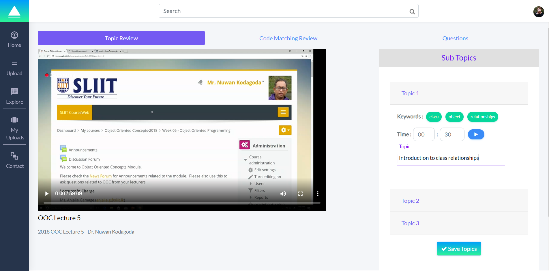
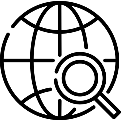


Figure 2‑13: VTutor video review UI

### Automatic Question Generation

Automatic question generation for video lecture is an isolated component from other components in the platform. Question generation process can divide into 3 major sub components. These sub components are explained in section wise in this section. High level system diagram of the automatic question generation component is shown in the Figure 2.13: High-level diagram of automatic question generation process

Main subject of the lecture



Web scraper to search and extract web content

Question generation

Present questions to end user

Search results and web content

Figure 2‑14: High-level diagram of automatic question generation process

#### Extracting Input Texts

Question generation process is mainly carried out using a rule-based methodology using text in paragraphs or sentences that contain factual statements as the input to the system. The first and the obvious approach for selecting the input text was the transcript of the video. However, this approach had to be discarded to several reason mentioned below since these recorded video lectures are done in a traditional classroom environment which can be longer than 1 hour to 2 hours.

* During a longer lecture, lecturer could talk about non-subject related content to take the focus of the students.
* Lectures carried out in a verbal manner that facts are scattered into several sentences.
* Lectures can be noisy.

To address the above issues, source of input text for the system had to chosen from internet rather than using the transcript of videos. To extract information or text from internet a search key phrase correlated with the lecture that could identify as the main discussion subject of the lecture needed to feed as an input to the system. For this key phrase extraction, systematic way of key phrases extraction had to be discarded since an error of this step will affect the outcome of the question generation process (error in a key phrase extraction could result in generating a set of question items that does not have any relation to the lecture video). To have 100% accuracy of this step, a basic human user input was taken as the key phrase or subject discussed in lecture.

Extracting content on internet is also carried out in two approaches. First approach is using a general web search from the key phrase to obtain articles or web pages.

* Once the key phrase is obtained, question generation process will systematically carry out a web search for the given key phrase. For this step, a search engine scraper implemented using GoogleScraper[[8]](#footnote-9) that imitate a web search query for the key phrase obtained and return the URLs of first 10 search results on a given search engine. Search engine of the scraper is set as Bing[[9]](#footnote-10) since it is the least problematic search engine that can be used for scraping. These extracted URLs are sent to another implementation of general web scraper implemented using Scrapy[[10]](#footnote-11) which will crawl through given URLs and extract text elements from web pages. However, using a general web search results to extract content has a possibility of copyright issues.
* To avoid this problem second approach is used where only the Wikipedia[[11]](#footnote-12) articles are used as a source to extract content. This method is also carried out in a similar manner explained in above section. Main difference in this approach is search engine query is specified to search using Wikipedia articles and content extraction process is carried out using Wikipedia-API[[12]](#footnote-13) instead of Scrapy implementation.

#### Question Generation

Question generation process for the platform is heavily based on the work of Michael Heilman’s Automatic Factual Question Generation from Text [22] which is considered by many researchers as one of best works regarding the automatic question generation using text and rule-based approach. In the system introduced by Heilman, question generation is carried out in 3 stages.

###### **Stage 1 *– NLP Transformation on Source Sentences***

In this stage extracted text from the web undergoes series of processes to obtain simplified factual statements from complex sentences that can be used to create question items.

For the simplified factual statement extraction, algorithm introduced in Michael Heilman and Noah A. Smith’s Extraction Simplified Statements for Factual Question Generation [50] is used. The extraction is based on semantic entailment (by removing discourse markers and adjunct modifiers and splitting conjunctions) and extraction by presupposition as explained in the paper. A high-level pseudocode of the algorithm is mentioned below which works on Penn Treebank style [51] phrase structure trees. To obtain the tree structure Stanford Parser [52][[13]](#footnote-14) is used.

**Algorithm 1 extractSimplifiedSentneces(t)**

*Tresults*← ∅

*Textracted*← {t} ∪ **extract** new sentences trees from *t* following: non-restrictive appositives; non-restrictive relative clauses; subordinate clauses with a subject and finite verb; and participial phrases that modify noun phrases, verb phrases, or clauses.

**for all** *t ∈ Textracted* **do**

*Textracted ←* *Tresults* ∪ **extractHelper(***t***)**

**end for**

**return** *Tresults*

**Algorithm 2 extractHelper(t)**

*Tresults* = ∅

**move** any leading prepositional phrases and questions in *t* to be the last children of the main verb phrase.

**remove** the following from *t:* noun modifiers offset by commas (non-restrictive appositives, non-restrictive relative clauses, parenthetical phrases, participial phrases), verb modifiers offset by commas (subordinate clauses, participial phrases, prepositional phrases), leading modifiers of the main clause (nodes that precede the subject).

**if** *t* has S, SBAR, or VP nodes conjoined with a conjunction c ∉ { or, nor } **then**

*Tconjuncts ←* **extract** new sentence trees for each conjunct in the leftmost, topmost set of conjoined S, SBAR, or VP nodes in *t.*

**for all** *tconjunct* ∈ *Tconjuncts* **do**

*Tresults ← Tresults* ∪ **extractHelper(***tconjunct***)**

**end for**

**else if** *t* has a subject and finite main verb **then**

*Tresults ← Tresults* ∪ {t}

**end if**

**return** *Tresults*

###### ***Stage 2 – Question Creation***

In this stage, the factual statements extracted using above algorithm is used as input to create possible question items. Creation of question is based on rules of wh-movement comes in linguistics. In Heilman’s work of question creation, he has introduced 6 steps that have to applied to create a question from an extracted factual statement which could contain noun phrase, prepositional phrase or subordinate clause as the answer phrase. Figure 2.14: Question creation process by transforming sentences illustrate the process of transforming a sentence to create a question using these 6 steps.

These 6 steps are explained briefly under this section. A deep explanation of these steps can be found in the thesis publication of Automatic Factual Question Generation from Text.

Figure ‑: Question creation process by transforming sentences

Decomposition of main verb

Input sentence

Mark Unmovable Phrases

Generate Possible Question Phrases

Subject-Auxiliary Inversion

Remove Answer, Insert Question Phrase

Post Processing

Question

**Marking Unmovable Phrases**

Mark the phrases of input tree as unmovable on the constraints of wh-movement. Tregex[[14]](#footnote-15), which is a Java program for identifying patterns in trees is used in this step.

**Generate Possible Question Phrases**

Input sentences are annotated with an entity types that each word token is labeled (such as PERSON, ORGANIZATION, LOCATION, etc.) using a Java implementation of Supersense Tagger described by Massimiliano Ciaramita and Yasemin Altun [53].

**Decomposition of the Main Verb**

Convert question phrases that does not contain an auxiliary verb into a form of *do* and base form of the main verb. This step is skipped if an auxiliary verb is already available in the sentence.

**Subject-Auxiliary Inversion**

Move auxiliary verb in sentence to a position in front the subject of the main-clause.

**Removing Answer Phrase and Inserting Question Phrase**

Removes the selected answer phrase from the sentence and insert question phrase generated using the selected answer phrase.

**Post-processing**

Additional post-processing to properly format question such as adding question marks to the end of question phrases, etc.

###### ***Stage 3 – Question Ranking***

In addition to question creation process Heilman’s paper introduce a question ranking component to select the most suitable questions. This stage is skipped in this implementation of platform.

#### Question Reviewing and Presenting to the User

Question items generated as described in the previous section will be saved to the metadata instance created initially by uploading the lecture video and other supporting materials. After the completion of all processes related the video lecture, lecturer will be notified as *video lecture processing is finished*. These processed videos will be shown to the lecturer in a separate section in the platform as *videos up for publish* where he or she can review the automatically generated components such as questions.

For the question generation component, questions will be shown with answers to the lecturer as a list. From this list questions can be removed or edited if they are invalid or does not match with the content of lecture. Answer of the question also can be edited and add distractors according to the lecturer’s will. A sample user interface of this process is shown in Figure 2.15: User interface question review

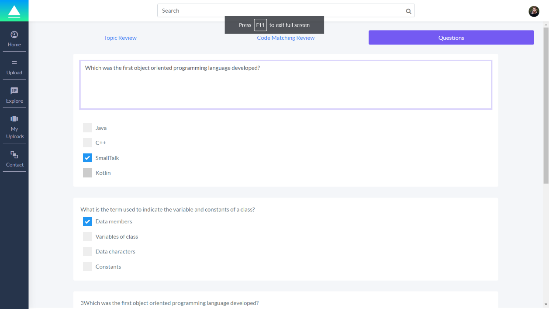


Figure 2‑16: User interface question review

### Slide Matching

Electronic slides are often added to Learning management Systems (LMS) together with the Recorded lecture videos. In the present, the platforms that is used do not provide the users the privilege to match a certain electronic slide with the lecture video in order to find the point of time in the video where the selected slide is discussed. Therefore, having a methodology to navigate through the lecture video using the electronic slides as an index can considerably improve the learnability and the ability to search while making the learning process efficient and easier.

As a platform that is developed to address the gap between the increasing requirements of the learners and existing products in the same domain, VTutor has provide a tool for the learners to match the electronic slides with the relevant lecture video. Therefore, once the user selects a slides from the slide deck displayed, the platform will automatically take the user to the point of time in the lecture video, where the selected slide is being discussed.

#### Methodology

Below are the steps that the platform mainly performs in order to provide the above discussed functionality.

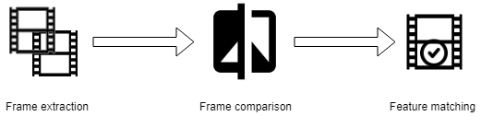


Figure 2‑17: Slide Matching Process Overview

*Frame extraction*

The lecture video generates frames at a rate of one frame per every second using Ffmpeg, a free and open source library for handling audio, video and other multimedia types. Each frame generated in this step is named in such a way that it has the time of occurrence (in seconds) with the name of the frame.

*Frame Comparing*

The user-selected slide is read and compared against all the frames of the lecture video generated in the previous step and the difference between every frame and the selected slide (as a percentage) is calculated using OpenCV2, an open source library for image processing and computer vision.

*Slide Matching*

As a result of the slide comparison carried out in the previous step, the algorithm chooses the first frame having less than 1% of difference with the compared slide and recognizes it as the first occurrence of the selected slide in the lecture video.

In the final step, after identifying the first frame with a difference less than 1% through the algorithm, the platform automatically navigates the lecture video to the point of time where that particular frame was occurred with the aid of the naming convention used when extracting frames using Ffmpeg in the first step.

#### Algorithm

Below mentioned is the algorithm that is used for frame comparison in the platform.

Given: The selected slide *m*, frames generated for the lecture video *frames*,

FOR frame in frames:

frame\_read = cv2.imread(frame, CV\_LOAD\_IMAGE\_GRAYSCALE);

slide\_read = cv2.imread(m, CV\_LOAD\_IMAGE\_GRAYSCALE);

res = cv2.absdiff(frame\_read, file\_read);

res = res.astype(np.unit8);

diff\_precentage = (numpy.count\_nonzero(res) \* 100) / res.size;

IF diff\_precentage < 1 THEN

OUTPUT frame;

break;

END IF

END FOR

In the process of selecting an appropriate algorithm for comparing the selected slide with the generated frames, several existing algorithms were taken into consideration such as Scale Invariant Feature Transform (SIFT), Speed Up Robust Features (SURF), Oriented Fast and Rotated BRIEF (ORB).

*Scale Invariant Feature Transform (SIFT)*

This algorithm has 4 basic steps.

1. Estimate a scale space extrema using the difference of Gaussian (doG).
2. Key-point localization where the key point candidates are localized and refined by eliminating the low contrast points.
3. Key point orientation assignment based on local image gradient
4. Descriptor generator to compute the local image descriptor for each key point based on the image gradient magnitude and orientation.

Although the algorithm produces accurate results, it is mathematically complicated and computationally heavy. As the algorithm is based on the histogram of gradients. Therefore, the gradients of each pixel in the patch need to be computed and these computations cost time. In addition, the algorithm cannot be performed on low powered devices and due to that, if used, the system would not be effective on low powered devices. [25]

*Speed Up Robust Features (SURF)*

The Speed Up Robust Features (SURF) algorithm approximates the difference of Gaussian (doG). Instead of Gaussian averaging, approximation of squares are used as the convolution of images are faster when integral images are used. This algorithm uses a BLOB detector to find the matching points. For feature description, the algorithm uses wavelet responses. It selects and area around the selected point, divides it further. For each region the wavelet response is taken and the SURF feature descriptor is taken. The sign of the Laplacian which is comuted is used for underlying points of interests and it can separately identify the bright BLOBs on the dark backgrounds from the reverse case. The feature comparison is only done if a common type of contrast is identified. SURF algorithm is faster than SIFT, but it is computationally heavy and needs a high powered devices to work on. In addition, the algorithm is patent protected. [25]

*Oriented Fast and Rotated BRIEF (ORB)*

ORB is a combination of FAST key point detector and BRIEF descriptor with some modifications. It uses the FAST key point detector to determine the key points, and the Harris corner measure is applied to the find the top n points identified by the FAST key point detector. It computes the intensity weighted centroid of the patch with located corner at the center. The direction of the vector from this corner point to centroid gives the orientation. Moments are computed to improve the rotation invariance. The descriptor BRIEF poorly performs if there is an in-plane rotation. In ORB, a rotation matrix is computed using the orientation of patch and then the BRIEF descriptors are steered according to the orientation. When considering the ORB algorithm for image matching its robustness is less than the other available algorithms. [25]

As a result of the analysis done to identify the suitable approach for image comparison, the top available algorithms displayed drawbacks that would negatively affect the performance and the usability of the platform. With the intension of providing a better user experience and increased performance, the VTutor platform uses an algorithm which is created for the platform specifically, simple yet accurate. The algorithm doesn’t high powered devices or specific requirements.

Using the algorithm, the platform is only able to build the synchronization between screen-recorded lecture videos and electronic lecture slides at the current state. Building the synchronization of electronic lecture slides with speaker-embedded lecture videos and other common type of lecture videos remain a constraint when interacting with the platform at the moment and is expected to address in the future releases.

An example for the slide matching scenario is shown below in figure 2.17.

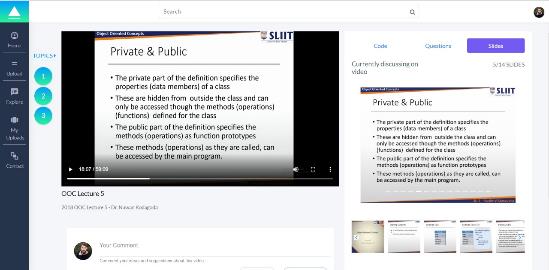


Figure 2‑18: User Interface for Slide Matching

#### Audio Denoising

A raw lecture video that is recorded with the speaker using a normal microphone usually contains a significant amount of noise that affects the quality of the audio like students chatter, breathing noise of the speaker and various other background noises produced by different sources. These noises become a disturbance for the users when watching the lecture videos as the background noises that is contained in the video makes the voice of the speaker unclear and even sometimes the voice of the speaker may not be heard due to the background noise. Such noises disturb the concentration on the video and negatively affect the learning process as well. Therefore, in VTutor platform with the intension of addressing this issue, the lecture video is automatically de-noised once it is uploaded, before making it available for the users to access. In order to address the above issue, the VTutor platform makes use of the deep learning de-noising API.

Once the raw audio that is extracted from the noised video is sent to the API through a POST request, the Deep Affects audio de-noising API removes noise from the audio signal and returns the de-noised audio clip to the web hook URL passed. The API is organized around REST and all the requests are made over SSL. All the requests and responses including errors are encoded in JSON. Among all the de-noising APIs available, Deep Affects has a large free tier which can handle 100 requests per day for a single user, and 5 requests can be made per minute to the API. It has reasonable pricing when the free tier is exceeded.

## Commercialization Aspects of the Product

As shown in the literature survey and research gap, although some of the available commercial products provide similar features to those introduced in our platform, none of them provided the facility of automating the video enhancement process. The V-Tutor platform has promising business potential in the field of eLearning. A Few of the business models available for the platform are mentioned below.

* Standalone platform for video lectures provided as a SaaS product, where customers can obtain a license and use the features.
* A Free platform for video lectures with premium content that viewers must pay purchase or subscribe to view.
* Integration with existing MOOCs and LMSs as a plugin or API.

Figure 2‑19 shows the business model canvas for the V-Tutor platform.

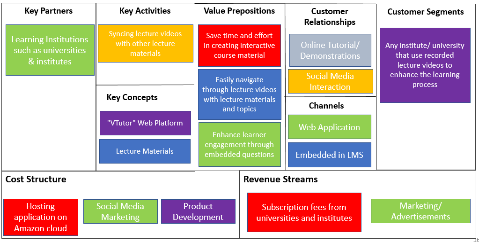


Figure 2‑19- Business model

## Testing and implementation

The frontend client of the platform is implemented as a web application developed using Angular 7 framework. Backend of the application is developed as a NodeJS Express REST application.

Add implementation of each component?

Question generation component is implemented as a separate web service using RESTful architectural pattern to increase the performance on the overall system and question generation component itself. Web scraping and text extraction processes are implemented using Python 3 and question generation process is implemented as a Java spring boot API.

# Results and discussion

## Results

This section presents a discussion of the results obtained for each individual component in the platform.

### Results for the Code Matching Component

For training the frame classification model, the same dataset of 900 images divided into the classes, “code” and “not code” was used. All training was performed without GPU acceleration on a laptop with 8GB of RAM and an intel core i5 8250u processor. Each model was trained for 20 epochs with 80 training steps per epoch and a batch size of 8. Table 3.1 compares the results obtained in training the 3 models on the dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Training time | Top 1 Validation accuracy | Top 1 Validation loss |
| VGG16 | 104s per epoch | 0.9387 | 0.1598 |
| Resnet50 | 26s per epoch | 0.8924 | 0.3351 |
| InceptionV3 | 96s per epoch | 0.8096 | 0.4187 |

Table 3.1- Training result comparison of candidate models

Figure 3‑1 compares the accuracy and loss curve for the InceptionV3 model. According to the graph, the model starts overfitting from the 3rd epoch onwards leading to a higher accuracy but also a high loss. Figure 3‑2 compares the accuracy and loss curve for the Resnet50 model during training. It performs marginally better in terms of generalization and does not seem to overfit the data. Finally Figure 3‑3 shows the accuracy and loss graphs for the VGG16 model. This model has proved to be the best out of the three models for this particular task as it has the highest accuracy as well as the lowest validation loss while not overfitting the data.

A close up of a map

Description automatically generated

Figure 3‑1- InceptionV3 comparison

A close up of a map

Description automatically generated

Figure 3‑2- Resnet50 comparison

A close up of a map

Description automatically generated

Figure 3‑3 - VGG16 comparison

### Results for Automatic Question Generation Component

A performance testing for the system was carried out using a personal computer with Intel Core i5 1.7GHz mobile processor and 8GB of RAM which use Windows as the operating system. Wikipedia article on C++ classes[[15]](#footnote-16) was used as the input text source for the system. According the Count Wordsworth[[16]](#footnote-17) website, Wikipedia article contains 186 sentences and 3851 words.

Question generation as an isolated system took ***21 minutes*** to parse, extract and generate question items from the given Wikipedia article. From the given 186 sentences system was able to create ***325 questions*** which doubled the question per sentence count.

Qualitative testing for the generated question items was carried out by providing a sample of 100 questions generated to 3 students studies in final year BSc. in Information Technology. The results of the qualitative testing are shown in Table 3.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rating** | Valid | Vague | Grammar Error | Wrong Answer |
| **Percentage** | 50% | 15% | 12% | 23% |

Table 3.2: Qualitative test results for question items generated in automatic question generation component

### Topic Modelling and Segmentation

Transcribing lecture video as accurately as possible is a key feature that will affect the accuracy of the topic model as well as text segmentation process. As discussed in the earlier section of this document lecture videos are transcribed using AsemblyAI. Table 3.2**.** shows the average confidence level of transcribing raw mp4 lecture videos, without any editing or noise removal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Video Name** | **Video Duration (hh:mm:ss)** | **Approx. Duration for transcribing (minutes)** | **Average confidence of the transcript** |
| OOC Lecture 5 | 00:59:09 | 17 | 0.8543 |
| OOC Lecture 6 | 01:01:11 | 23 | 0.8231 |
| OOC Lecture 7 | 00:59:27 | 19 | 0.8711 |
| OOC Lecture 9 | 00:57:31 | 14 | 0.8972 |

Table 3.3 Word confidence level from AssemblyAI

Apart from the topic tiling algorithm used in this research for text segmentation, there are various other algorithms capable of segmenting text into subtopics. Pk value is a metric that is popularly used for evaluating text segmentation algorithms. It based on the window size and the segmentation boundaries of the algorithm. Lower the Pk value, better the performance of the algorithm. Table 3.3 shows a list of Pk values for Choi data set for different text segmentation algorithms in literature when number of segments are provided and unprovided.

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Segments** | |
| **Provided** | **Un provided** |
| TextTiling (TT) | 44.48 | 49.51 |
| C99 | 11.20 | 12.73 |
| U00 (Utiyama and Isahara, 2001) | 9 | 10 |
| F04 (Fragkou et al., 2004) | 5.39 |  |
| M09 (Misra et al., 2009) | 2.73 |  |
| C99LDA | 2.69 | 3.24 |
| TTLDA | **1.04** | 2.89 |
| Topic Tiling | 1.06 | **1.39** |

Table 3.4 List of lowest Pk values for the Choi data set for different algorithms.

Source : <https://pdfs.semanticscholar.org/f731/ad6b5aa5ca61ed61e64d3faf94a4b78b22f3.pdf>

As seen in the table, topic tiling algorithm outperforms all other algorithms when the number of segments are not given. Hence it is the best algorithm for segmenting text based on topic when compared with other existing algorithms.

Table 3.4, Table 3.5 and Table 3.6shows the results of topic tiling algorithm for three sample videos compared to manual segmentation of the same video in a human perspective in two scenarios: number of segments provided and number of segments not provided.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Video Title - OOC Lecture 5** | | | | |
| **Number of segments not provided** | | | | |
|  | **Topic Tiling Algorithm** | | **Manual Segmentation** | |
| **No. of segments** | 5 | | 3 | |
| **Number of segments provided** | | | | |
|  | **Timestamps (hh:mm:ss)** | | | |
| **Using Algorithm** | **Manual Segmentation** | | **Time difference in minutes** |
| **Segment 1** | 00:04:45 | 00:02:33 | | 2.2 |
| **Segment 2** | 00:17: 54 | 00:19:31 | | 1.62 |
| **Segment 3** | 00:42:11 | 00:39:28 | | 2.71 |

Table 3.5 Sample 1 text segmentation results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Video Title - OOC Lecture 6** | | | | |
| **Number of segments not provided** | | | | |
|  | **Topic Tiling Algorithm** | | **Manual Segmentation** | |
| **No. of segments** | 8 | | 5 | |
| **Number of segments provided** | | | | |
|  | **Timestamps (hh:mm:ss)** | | | |
| **Using Algorithm** | **Manual Segmentation** | | **Time difference in minutes** |
| **Segment 1** | 00:02:10 | 00:03:37 | | 1.45 |
| **Segment 2** | 00:10:40 | 00:08:32 | | 2.14 |
| **Segment 3** | 00:16:12 | 00:13:43 | | 2.48 |
| **Segment 4** | 00:25:56 | 00:26:34 | | 0.64 |
| **Segment 5** | 00:53:39 | 00:50:22 | | 3.28 |

Table 3.6 Sample 2 text segmentation results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Video Title - OOC Lecture 6** | | | | |
| **Number of segments not provided** | | | | |
|  | **Topic Tiling Algorithm** | | **Manual Segmentation** | |
| **No. of segments** | 7 | | 5 | |
| **Number of segments provided** | | | | |
|  | **Timestamps (hh:mm:ss)** | | | |
| **Using Algorithm** | **Manual Segmentation** | | **Time difference in minutes** |
| **Segment 1** | 00:03:42 | 00:02:37 | | 1.03 |
| **Segment 2** | 00:15:34 | 00:18:30 | | 2.93 |
| **Segment 3** | 00:27:12 | 00:25:58 | | 1.23 |
| **Segment 4** | 00:37:50 | 00:36:28 | | 1.30 |
| **Segment 5** | 00:46:10 | 00:42:59 | | 3.19 |

Table 3.7 Sample 3 text segmentation results

### 3.1.3. Results for Slide Matching Component

3.1.3.1.Measuring performance

Below is a comparison of the SIFT, SURF, ORB and the algorithm used in the vTutor platform in terms of performance. All four algorithms were tested in a scenario where they had to read the 100 images to find the exact match to the given image (m) and the time taken by each algorithm is measured (in seconds).

|  |  |
| --- | --- |
| Video Title – OOC Lecture 5 |  |
| Algorithm | Time taken to find the match(in seconds) |
| Manual Matching | 47 |
| SIFT | 6 |
| SURF | 8 |
| ORB | 11 |
| Our solution | 8 |

Table 3.7: Performance comparison - OOC Lecture 5

|  |  |
| --- | --- |
| Video Title – OOC Lecture 6 |  |
| Algorithm | Approximate time taken to find the match(in seconds) |
| Manual Matching | 32 |
| SIFT | 6 |
| SURF | 9 |
| ORB | 11 |
| vTutor | 8 |

Table 3.8: Performance comparison - OOC Lecture 6

By comparing the results, it is evident that the algorithm used in the discussed platform has a considerable efficiency when compared with other algorithms. SIFT having the least time to find the match cannot be used in this context as it requires high powered devices to perform efficiently.

3.1.3.2. Measuring accuracy

Below is the comparison of the SIFT, SURF, ORB and the algorithm used in vTutor platform in terms of accuracy. All four algorithms were tested in a scenario where they had to read and identify the matching frame from all the frames generated by a given video and the accuracy of the match is measured. Table 4 is produced with the results calculated using the accuracy of four tests conducted.

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| SIFT | 99% |
| SURF | 99% |
| ORB | 95% |
| vTutor | 98% |

Table 3.8: Accuracy Measure

## Discussion

When considering the base models used to create the binary image classifier, VGG16 outperformed Resnet50 and InceptionV3. This could be attributed to the fact that VGG16 is a relatively simple model which requires very little tuning to work properly. The fact that the InceptionV3 based model started overfitting leads to the conclusion that more finetuning is required to get a viable result. For example, instead of freezing all convoluted layers, some of the deeper layers could remain trainable. Furthermore, regularization by using dropout layers during training is another course of action which can reduce overfitting. The sub-optimal results may also be due to the fact that the dataset was generated manually in a short period of time.

The performance results of the question generation component in can be vary due to several reasons as mentioned below.

* Complexity of the input text source
  + Providing complex input text source to the system can increase the time taken to simplify and extract factual statements from it which could results a delay in overall question generation process.
  + However, selecting a less complex input text source is not under control of the system in the current implementation of the platform.
* Performance of the computer
  + Some components of the system are high memory intensive (such as parsing the input text source). Increasing the performance of the computer can result in an increase of the system.
  + In a hosted environment system could work faster. However, server costs can be high.

Quality of the question items generated mainly based on the quality of input sentences. Using web scraped content on different subjects can vary this quality in an uncontrollable manner. To increase the quality of questions in future works, providing the input text source as supporting materials to the platform (reference textbooks, etc.) can increase the quality of the question items. Implementing a ranking system as mentioned in Heilman’s work could also increase the quality of output.

MCQ distractors are not generated from the question generation component. In future works, an implementation of automatic distractor generation will be an advantage for the lecture creators.

As discussed in section 2.1.3 usage scraped web content to generate questions can result in a violation of terms of use in some websites. Hence the decision to use only Wikipedia articles for scraping process is made.

When considering about transcribing lecture videos, according

to the figures in Table 3.2, transcribing a video using AssemblyAI takes around 20-30% of the duration of the video, which is tolerable delay because all the processing of the videos are dome in the backend and user do not have to wait until the videos is processed. Once a user uploads lecture materials through the VTutor web app it is automatically subjected for processing including transcription extraction and topic modelling. On successful completion of processing the user (lecturer) will notified through the web app and the users can then review the processed video, edit and publish for students to view.

Furthermore most of the transcripts obtained using AssemblyAI have a confidence level of approximately between 0.8 and 0.9, and the major factor which affects the drop in confidence level is the accent and the background noises. All the videos subjected for testing is done by Sri Lankan lecturers with an Asian accent and most of these lectures includes background noise such as student voices which is commonly seen in a typical lecture hall. When considering all these factors, it can be concluded that AssemblyAI is a good for our platform because it transcribes lectures with a fairly high level of confidence within a reasonable period of time.

A main contributing factor for text segmentation is how well the LDA model is trained. Since LDA is an unsupervised learning technique, lot of data is required to train the model effectively. Many of the available data sets on internet used for training LDA are not related to lecture video transcripts, instead many of these are based on news paper articles and other written text. None, these datasets are not suitable for training the LDA model in this context because unlike written text, transcripts contain spoken words which are spontaneous and less saturated with topic related words. Hence, one of the main challenges in this research is to train the LDA model effectively, due to lack of data. The results can be further improved by training the model more data.

# Conclusion

# References

|  |  |
| --- | --- |
| [1] | S. P. M. S. Ebrahim Karami, "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Disorted Images," Faculty of Engineering and Applied Sciences, Memorial University, Canada. |
| [2] | "Importance and Effectiveness of E-learning," 1 December 2015. [Online]. Available: https://higheredrevolution.com/importance-and-effectiveness-of-e-learning-9513046ed46c. [Accessed 6 March 2019]. |
| [3] | M. Brandsteidl, T. Mayerhofer, M. Seidl and C. Huemer, "Replacing traditional classroom lectures with lecture videos: an experience report," in *Proceedings of the 8th edition of the Educators' Symposium*, 2012. |
| [4] | M. Maher, H. Lipford and V. Singh, "Flipped classroom strategies using online videos," UNC Charlotte, North Carolina, 2013. |
| [5] | M. Chau, M. Lin, J. F. Nunamaker and H. Chen, "Segmentation of Lecture Videos Based on Text: A Method Combining Multiple Linguistic Features," in *Proceedings of the 37th Hawaii International Conference on System Sciences*, Hawaii, 2004. |
| [6] | Z. Woolfitt, "The effective use of video in higher education," 2015. [Online]. Available: https://www.inholland.nl/media/10230/the-effective-use-of-video-in-higher-education-woolfitt-october-2015.pdf. [Accessed 6 March 2019]. |
| [7] | M. J. Rubin, "The effectiveness of live-coding to teach introductory programming," in *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE ’13*, Denver, Colorado, USA, 2013. |
| [8] | V. K. Kamabathula and S. Iyer, "Automated Tagging to Enable Fine-Grained Browsing of Lecture Videos," in *2011 IEEE International Conference on Technology for Education*, 2011. |
| [9] | Luca Ponzanelli et al, "CodeTube: Extracting Relevant Fragments from Software Development Tutorials," in *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE ’16*, 2016. |
| [10] | A. M. a. M. L. L. Ponzanelli, "StORMeD: Stack Overflow ready made data," in *Proceedings of MSR 2015 (12th Working Conference on Mining Software Repositories)*, 2015. |
| [11] | J. Ott, A. Atchison, P. Harnack, A. Bergh and E. Linstead, "A deep learning approach to identifying source code in images and video," in *Proceedings of the 15th International Conference on Mining Software Repositories - MSR ’18*, 2018. |
| [12] | K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR,* vol. abs/1409.1556, 2014. |
| [13] | K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. |
| [14] | C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. |
| [15] | M. Lin, C. B. R. Diller, Ming, N. Forsgren, Y. Huang and J. F. Nunamaker, Jr, "Segmenting Lecture Videos by Topic: From Manual to Automated Methods," in *Eleventh Americas Conference on Information Systems*, Omaha, NE, 2005. |
| [16] | H. J. Zhang and S. W. Smoliar, "Developing power tools for video indexing and retrieval," in *SPIE'94 Storage and Retrieval for Video Databases*, San Jose, CA, USA, 1994. |
| [17] | X. Che, H. Yang and C. Meinel, "Lecture Video Segmentation by Automatically Analyzing the Synchronized Slides," in *MM’13*, Barcelona, Spain, 2013. |
| [18] | J. Allan, J. Carbonel, G. Doddington, J. Yamron and Y. Yang, "Topic detection and tracking pilot study: Final report," in *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*. |
| [19] | R. Shah, D. Shah and L. Kurup, "Automatic Question Generation for Intelligent Tutoring Systems," in *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, Mumbai, 2017. |
| [20] | Y.-T. Huang, Y.-M. Tseng, Y. S. Sun and M. C. Chen, "TEDQuiz: Automatic Quiz Generation for TED Talks Video Clips to Assess Listening Comprehension," in *2014 IEEE 14th International Conference on Advanced Learning Technologies*, Athens, 2014. |
| [21] | G. Erkan and D. R. Radev, "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization," in *Journal of Artificial Intelligence Research, Vol 22*, 2004. |
| [22] | M. Heilman, "Automatic Factual Question Generation from Text," Doctoral thesis, Carnegie Mellon University, Pittsburgh, 2011. |
| [23] | M. Heilman and N. A. Smith, "Question Generation via Overgenerating Transformations and Ranking," in *Technical report, Language Technologies Institute, Carnegie Mellon University Technical Report*, 2009. |
| [24] | A. Krishna, P. Bhowmick, K. Ghosh, A. Sahu and S. Roy, "Automatic Generation and Insertion of Assessment Items in Online Video Courses," in *IUI Companion '15 Proceedings of the 20th International Conference on Intelligent User Interfaces Companion*, Atlanta, Georgia, 2015. |
| [25] | M. Heilman and N. A. Smith , "GoodQuestion! StatisticalRankingforQuestionGeneration," in *HLT '10 Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Log Angeles, California, 2010. |
| [26] | M. Tosic and M. Cubric, "SeMCQ – Protégé Plugin for Automatic Ontology-Driven Multiple Choice Question Tests Generation," in *Procs of the 11th International Protege Conference*, 2009. |
| [27] | M. Al-Yahya, "OntoQue: A Question Generation Engine for Educational Assesment Based on Domain Ontologies," in *2011 IEEE 11th International Conference on Advanced Learning Technologies*, Athens, 2011. |
| [28] | A. Papasalouros , K. Kanaris and K. Kotis, "Automatic Generation Of Multiple Choice Questions From Domain Ontologies," in *IADIS International Conference e-Learning 2008*, Amsterdam, 2008. |
| [29] | F. Wang, C. W. Ngo and T. C. Pong, "Synchronization of lecture videos and electronic slides by video text analysis," in *Proceedings of the eleventh ACM international conference on Multimedia - MULTIMEDIA ’03*, Hong Kong, 2003. |
| [30] | "Zentation.com - Webinar software," Zentation.com, 2019. [Online]. Available: http://www.zentation.com/. [Accessed 08 March 2019]. |
| [31] | K. Hymavathy and P. Janardhanan, "Noise Filtering in Speech Using Frequency Response Masking Technique," *International Journal of Emerging Trends in Engineering and Development,* 2013. |
| [32] | S. Muangjaroen and T. Yingthawornsuk, "A Study of Noise Reduction in Speech Signal Using FIR Filtering," in *Proceedings of the International Conference on Advances in Electrical and Electronics Engineering*, Pattaya, 2012. |
| [33] | T. L. Kumar, K. S. Rajan and ., "Noise Suppression in Speech Signals Using Adaptive Algorithms," *International Journal of Engineering Research and Applications,* vol. 2, pp. 718-721, 2012. |
| [34] | R. S. J. G. V. R. S. T. M. a. K. A. Aggarwal, "Noise Reduction of Speech Signal Using Wavelet Transform with Modified Universal Threshold," *International Journal of Computer Applications,* vol. 20, pp. 15-19, 2011. |
| [35] | E. Verteletskaya and B. Simak, "Speech Distortion Minimized Noise Reduction Algorithm," in *Proceedings of theWorld Congress on Engineering and Computer Science*, San Francisco, 2010. |
| [36] | M. Karam, H. F. Khazaal, H. Aglan and C. Cole, "Noise Removal in Speech Processing Using Spectral Subtraction," *Journal of Signal and Information Processing,* vol. 5, no. 2, pp. 32-41, 2014. |
| [37] | "Audacity - Free Open source, cross platform audio software," Audacity, 2019. [Online]. Available: https://www.audacityteam.org/. [Accessed 05 March 2019]. |
| [38] | J. Brownlee, "A Gentle Introduction to Transfer Learning for Deep Learning," Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/transfer-learning-for-deep-learning/. [Accessed 1 March 2019]. |
| [39] | W. Nowak, "How to Train Your Model (Dramatically Faster)," Towards Data Science, 2019. [Online]. [Accessed 1 March 2019]. |
| [40] | J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, p. 888–905, 2000. |
| [41] | "Image Preprocessing - Keras Documentation," Keras.io, 2019. [Online]. Available: https://keras.io/preprocessing/image/. [Accessed 05 Aug 2019]. |
| [42] | J. Yosinski, J. Clune, Y. Bengio and H. Lipson , "How transferable are features in deep neural networks?," in *27th International Conference on Neural Information Processing Systems*, Montreal, Canada, 2014. |
| [43] | Xlinux.nist.gov, "Ratcliff/Obershelp pattern recognition," 2019. [Online]. Available: https://xlinux.nist.gov/dads/HTML/ratcliffObershelp.html. [Accessed 02 Aug 2019]. |
| [44] | "7.4. difflib — Helpers for computing deltas — Python 2.7.16 documentation," 2019. [Online]. Available: https://docs.python.org/2.7/library/difflib.html#module-difflib. [Accessed 12 Aug 2019]. |
| [45] | T. Doll, "LDA Topic Modeling: An Explanation," 25 June 2018. [Online]. Available: https://towardsdatascience.com/lda-topic-modeling-an-explanation-e184c90aadcd. [Accessed 4 August 2019]. |
| [46] | A. Onan, S. Korukoglu and H. Bulut, "LDA-based Topic Modelling in Text Sentiment Classification: An Empirical Analysis," *International Journal of Computational Linguistics and Applications,* vol. VII, pp. 101-119, 2016. |
| [47] | "Latent Dirichlet allocation," wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Latent\_Dirichlet\_allocation. [Accessed 4 August 2019]. |
| [48] | "Topitiling," Github, [Online]. Available: https://github.com/riedlma/topictiling. [Accessed 4 August 2019]. |
| [49] | M. Riedl and C. Biemann, "How Text Segmentation Algorithms Gain from Topic Models," in *2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Montreal, Canada, 2012. |
| [50] | M. Heilman and N. A. Smith, "Extracting Simplified Statements for Factual Question Generation," 2010. |
| [51] | M. . P. Marcus, M. A. Marcinkiewicz and B. Santorini, "Building a large annotated corpus of English: the penn treebank," *Computational Linguistics - Special issue on using large corpora: II,* vol. 19, no. 2, pp. 313-330, June 1993. |
| [52] | D. Klein and C. D. Manning, "Fast Exact Inference with a Factored Model for Natural Language Parsing," in *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. |
| [53] | M. Ciaramita and Y. Altun, "Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger," in *EMNLP '06 Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, 2006. |
| [54] | S. P. M. S. Ebrahim Karami, "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Disorted Images," Faculty of Engineering and Applied Sciences, Memorial University, Canada, 2015. |

# Appendices

dsda

1. https://nodejs.org/en/ [↑](#footnote-ref-2)
2. https://angular.io/ [↑](#footnote-ref-3)
3. https://aws.amazon.com/s3/ [↑](#footnote-ref-4)
4. https://aws.amazon.com/sns/ [↑](#footnote-ref-5)
5. https://ffmpeg.org/ [↑](#footnote-ref-6)
6. https://github.com/tesseract-ocr/ [↑](#footnote-ref-7)
7. <https://www.assemblyai.com/> [↑](#footnote-ref-8)
8. <https://github.com/NikolaiT/GoogleScraper> [↑](#footnote-ref-9)
9. Bing.com [↑](#footnote-ref-10)
10. <https://scrapy.org/> [↑](#footnote-ref-11)
11. <https://www.wikipedia.org/> [↑](#footnote-ref-12)
12. <https://github.com/martin-majlis/Wikipedia-API/> [↑](#footnote-ref-13)
13. <https://nlp.stanford.edu/software/lex-parser.shtml> [↑](#footnote-ref-14)
14. <https://nlp.stanford.edu/software/tregex.shtml> [↑](#footnote-ref-15)
15. <https://en.wikipedia.org/wiki/C%2B%2B_classes> [↑](#footnote-ref-16)
16. <http://countwordsworth.com/> [↑](#footnote-ref-17)