AUTOMATIC KEYWORD DETECTION FOR TEXT SUMMARIZATION

A Thesis Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Raga Shalini Koka

May 2019

AUTOMATIC KEYWORD DETECTION FOR TEXT SUMMARIZATION

Raga Shalini Koka

APPROVED:

Dr. Jaspal Subhlok, Advisor

Dr. Thamar Solorio

Dr. Christopher Barr

Dr. Dan E. Wells, Dean,

College of Natural Sciences and Mathematics

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor, Dr. Jaspal Subhlok and Dr. Thamar Solorio for their valuable guidance and indelible encouragement throughout the course. Their support, help, and enthusiasm motivated and reinforced my confidence at every stage of my research. Our discussions provided me great insight to look through a new perspective conceptually and accomplish the goal of this thesis. I want to thank Dr. Christopher Barr for being part of the thesis and supporting me constantly. I also want to thank Dr. Shishir Shah and Dr. Ioannis Konstantinidis for their invaluable suggestions in various phases of the project. Lastly, I would like to thank my parents who were always my great support and encouragement to strive towards achieving my goal and making this work a success.

AUTOMATIC KEYWORD DETECTION FOR TEXT SUMMARIZATION

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Raga Shalini Koka

May 2019

ABSTRACT

Lecture videos are extremely useful and great learning companions for students. The ICS (Indexed, Captioned, and Searchable) video project provides students a flexible way to navigate across the lectures by automatically dividing the lecture into topical segments. Presenting keywords to every segment can provide an overview of the content discussed in a segment and improve navigation. Identifying keywords manually requires human effort and consumes a lot of time for lecture videos that are typically an hour or longer. This thesis proposes methods to automatically detect keywords to summarize the content in a video segment.

The input to the keyword detection algorithm is text from the video frames extracted by OCR, and I enhance the text with auto-correction in a post-processing pass. Automatically detecting keywords is challenging as the importance of a word depends on a variety of factors such as frequency, font size, and duration of time it is present on the screen. Other factors include relative frequency in a video segment versus the rest of the video and domain significance derived from external sources. This thesis explores how these factors contribute to the importance of a word and how they can be combined to identify good keywords.

I evaluated the performance of the proposed methods by comparing the keywords generated by the algorithm with the tags chosen by experts on 121 segments of 11 videos from different departments like Computer Science, Biology, and Biochemistry. I initialized the features to different combinations of weights and computed metrics like precision,

recall, F1, BLEU score, and correlation scores. I also presented an analysis of errors and different areas that can be explored to generate higher quality keywords.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1: INTRODUCTION

Many academic institutions are publishing lectures online to be accessed by the students. With the advancements in technology, Online Learning or E-learning has gained enormous popularity. The pervasive use of communication systems and devices that facilitate the creation and distribution of digital content [1] are significantly contributing to the success of Online Learning. Students can access these resources anytime, anywhere, to adapt to their learning style [2].

The usage of lecture videos has become a common trend among instructors and students. Students use these videos as autonomous resources, e.g., in the case of distance learning, or in conjunction with classroom lectures to supplement classroom teaching. The emerging popularity has led universities including the University of Houston, Stanford, and MIT, to publish their lectures online. Massively Open Online Courses (MOOC) such as Udemy, Coursera, and Pluralsight have been successful in delivering online lectures in different domains by enhancing the students' learning experience. Students can make use of these lectures to compensate for a missed class or review purposes. The availability of classroom lectures can be helpful to recreate the classroom experience and capturing the student-professor interaction.

A lot of ongoing research aims to ease the navigation of long lecture videos using 'Indexing,' which automatically divides a video into segments indicating different topics and 'Keyword Search,' that identifies video segments matching a particular keyword [3].

Although these features greatly save time in accessing the video content, the student would not be able to see the relevant keywords of a topic discussed in a video segment.

1.1 Motivation

The motivation for this thesis has developed from the aim to automatically present the relevant keywords or tags as a summary of each segment of a video. Although navigation inside a video has improved with indexing techniques, it would be difficult for students to guess the content present in a segment by viewing the first video frame of the segment. Presenting keywords would allow the students to glance at the content discussed in a segment and efficiently judge if they should review the topic or not, thus making the navigation even more quick and efficient. It is possible to manually identify the keywords for each segment in a video lecture with the help of instructor; however, performing this task manually is cumbersome as a typical lecture video is about 60-90 minutes and sometimes even longer. Also, the indexing of the video might change as new algorithms continue to evolve, improving the accuracy to identify index points. In this case, the instructor should rearrange all the keywords for the segments accordingly. So, manually identifying keywords is an expensive process. Automatic keyword detection would help in providing a faster and efficient way to tackle the problems by extracting words that are more relevant to the topic. Thus, identifying keywords automatically would be useful for faster navigation across a lecture video.

1.2 Summary of Research

The primary goal of this research is to develop methods that identify keywords for each segment of a lecture video. The keywords extracted can be phrases or *n-grams*. An n-gram is a sequence of 'n' terms. Initially, a long video is divided into multiple segments using indexing algorithms [3]. OCR technology detects the text on the segments, and the 'Keyword Detection Algorithm' takes the text as input. I outline the key considerations made by the algorithm to determine the keywords for the video segments below:

1. The text extracted by OCR contains noise. As detecting keywords on the noisy text may not produce good results, the text is post-processed by applying a two-phase auto-correction technique. The first phase uses suggestions from Google to reduce the noise in the text and the second phase substitutes rare, misspelled words by referring to the high-frequency words of the lecture.
2. The keyword detection algorithm extracts all the possible n-grams and validates them by searching the n-gram database from Google to retain the meaningful n-grams.
3. The algorithm performs pre-processing to eliminate the n-grams containing stop words and group them based on the stemming results.
4. Several factors such as frequency, inverse segment frequency, domain significance, font size, and duration determine the importance of an n-gram.
5. Frequency determines the importance of an n-gram in a segment, whereas inverse segment frequency reduces the effect of frequency if the n-gram widely occurs in

rest of the segments of a video. The keyword detection algorithm uses external dictionaries provided by Oxford to identify the n-grams containing words related to the lecture's domain and boost their scores.

6. The font size is another factor specifying the importance of an n-gram. The algorithm calculates the average font weight of each word in a segment and estimates the n-gram font score. Similarly, it computes the duration of time for which an n-gram occurs in a segment of the video.
7. Lastly, the keyword detection algorithm combines all the factors and generates a final score to each n-gram present in a segment. Then it ranks the n-grams based on the score and selects the top n-grams in each segment to present them as a summary to the users.

To evaluate the quality of keywords generated by the keyword detection algorithm, I developed a tool to collect the keywords for the video lectures from the instructors and students involved with the course. These manually assigned keywords are compared with the keywords generated by the algorithm and computed precision, recall, F1, BLEU, and correlation scores. I gathered keywords from experts for 121 segments of 11 videos from various departments of the College of Natural Sciences and Mathematics such as Computer Science, Biology and Biochemistry. The proposed methods achieve a precision of 54.5%, recall of 70.7%, and F1 score of 61.6%.

1.3 Thesis Outline

I organized the thesis as follows: The work presented here is part of a larger ICS (Indexed Captioned and Searchable) VideoPoints. Chapter 2 provides background information on the ICS Videos Project. Chapter 3 presents the related work done on keyword extraction and OCR correction. Chapter 4 elaborates the various pre-processing techniques and methods used to identify keywords from the video segments. It also elaborates on the methods used to rectify the OCR errors along with numerous significant challenges faced in this research. Chapter 5 explains the tool developed to collect ground truth from the users and process of evaluation along with the analysis of the results. It also discusses the reasons for errors generated by the system. Finally, Chapter 6 summarizes the current work and presents the potential future perspective of this work.

CHAPTER – 2: BACKGROUND – ICS VIDEO PROJECT

The ICS (Indexed, Captioned, and Searchable) Video Project is designed to make video lectures easily accessible to the students by providing index points (segments), captions and keyword search in a video lecture. It mainly aims to enhance the students' learning process by allowing to navigate and access the desired content easily. The main components of the ICS Videos are the 'Indexing module,' 'Captioning module,' and the 'Keyword Search module.'

The instructor records the classroom lecture by recording his/her computer screen while delivering a prepared viewgraph like PowerPoint in addition to the audio. The ICS server takes the uploaded video and automatically processes it by creating indexes, generating captions, and producing keywords to facilitate search. The keywords extracted in this process are words from the text produced by Optical Character Recognition (OCR) technology. All these words need not signify the content that has occurred in the video segment. They are mainly used to filter the segments of a video when the user searches inside the video. This thesis aims to extract keywords and present them to the user as a summary signifying the main content of the video segment. Note that this is different from identifying segments in a video when searching for a specific occurrence of a word.

Figure 2.1 shows a customized player built with sophisticated features like indexing, captioning and search. ICS Video Player is an HTML5 based player capable of streaming video over the Internet. The player consists of a playback component, index panel,

transcripts display panel and a search box [4]. The central portion of the player in the middle is the playback component. It has several options to play, pause, control the speed of the video, and hide index, and transcript panels. The bottom panel called the index panel represents different index points. The right side of the video contains interactive transcripts called the transcript display panel. There is a search box above the index panel which allows the user to search for index points containing the search term.

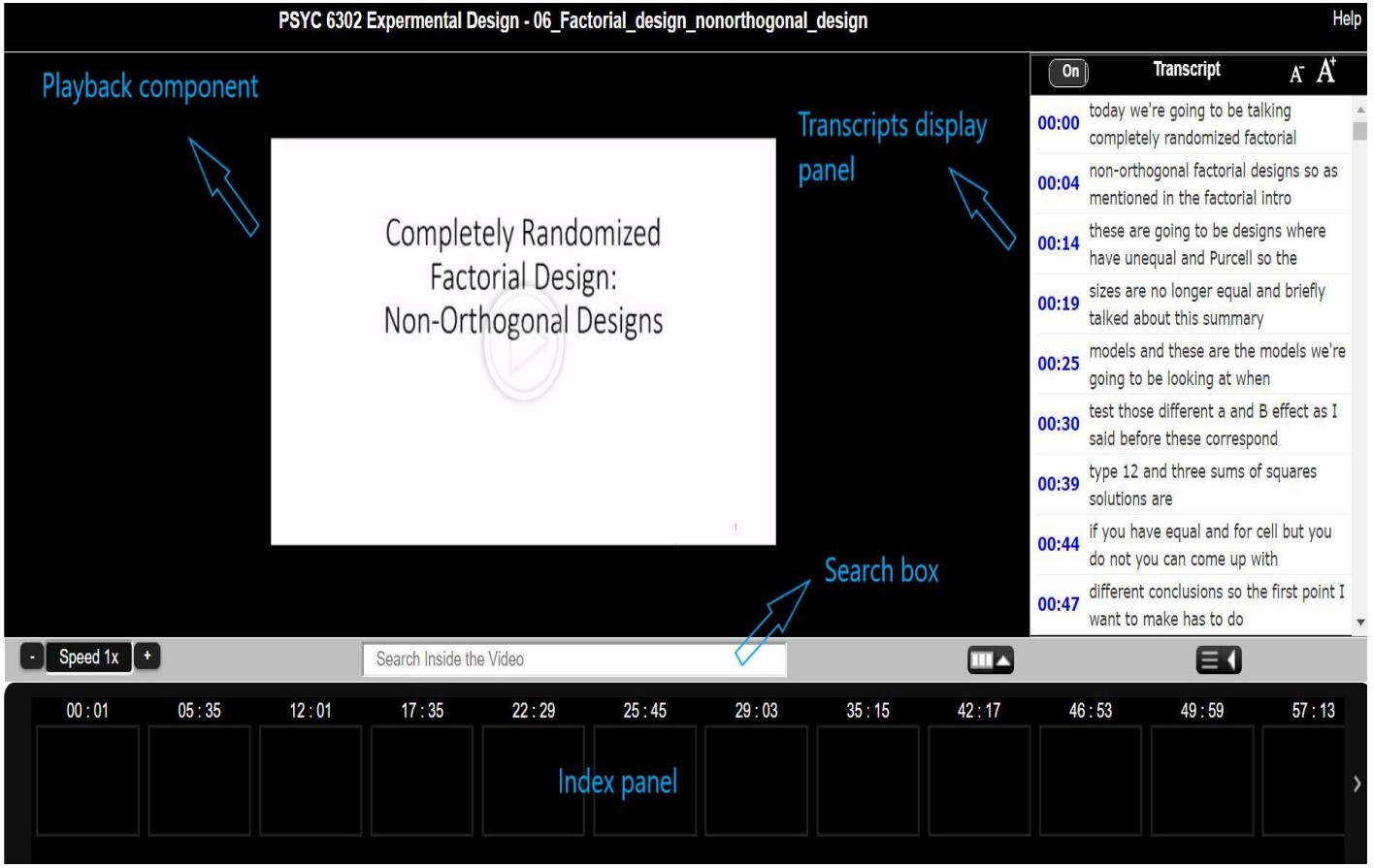

Figure 2.1 An Overview of ICS Player and its Components

2.1 Indexing

The process of dividing the video lecture into segments that represent different sub-topics is called ‘Indexing’ or ‘Segmentation.’ This algorithm detects significant scene changes in a video and marks these changes as *Transition Points*. *Index Points* are subsets of these transition points that represent a different sub-topic. The indexing algorithm selects the appropriate transition points as index points [5]. Figure 2.2 illustrates the transition points in a sequence of video frames and Figure 2.3 demonstrates the selection of index points from transition points. A segment is a set of transition points from one index point to another. In Figure 2.3, the first 4 transition points form a segment.

Figure 2.2 Transition Points in a Video. The 3rd frame is a New Transition Point [5]

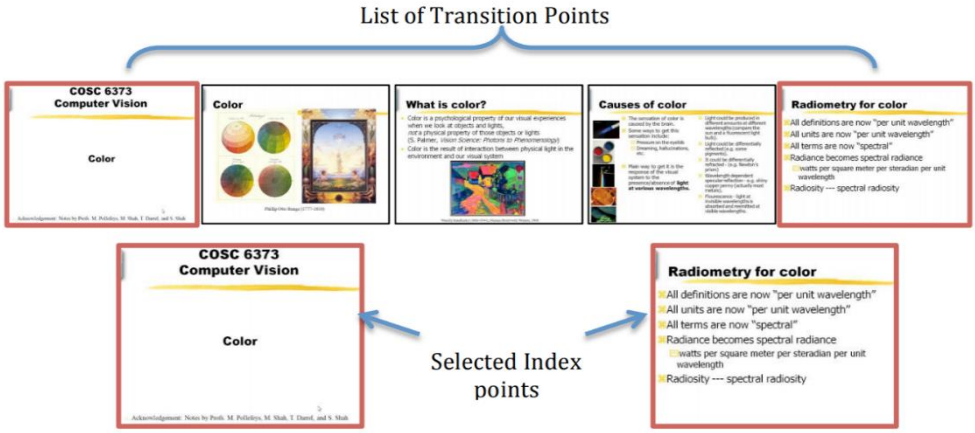

Figure 2.3 Selection of Index Points from Transition Points

The ICS system detects transition points in a video by comparing the RGB (Red, Green, Blue) values of the corresponding pixels in the two images for similarity. The system optimizes the process of comparison by employing a binary search mechanism to select frames at certain intervals. Image difference between the successive transition points forms the criteria for the selection of index points. The evaluations from the previous work showed that the index points were accurate most of the time but did not always represent a topic change [5].

2.2 Captioning

ICS Video Player has an in-built ‘Captioning’ feature. Captioning is used to enhance the accessibility of lecture videos. Captioning was primarily motivated to make the lectures available to deaf and improve the experience of hearing and foreign language students [5]. The video player displays the captions on the video screen along with the view graphs, and the player shows the complete transcript on the right-hand side. The ICS player provides a feature to turn on/off the captions and the transcripts. Although the ICS framework can generate captioning automatically, a certain degree of manual correction is desirable due to the limitations of Automatic Speech Recognition (ASR) tools.

2.3 Searching

The ‘Search’ feature in the ICS Video Player enables search inside a video. The process of identifying all the segments where a keyword exists is keyword search. The text box above the index panel in Figure 2.1 is used to perform the search.

The indexer creates the video segments as well as transition point frames. OCR detects the text on these frames and stores them in a database. The ICS Player loads the video along with the associated keywords on a playback request [5].

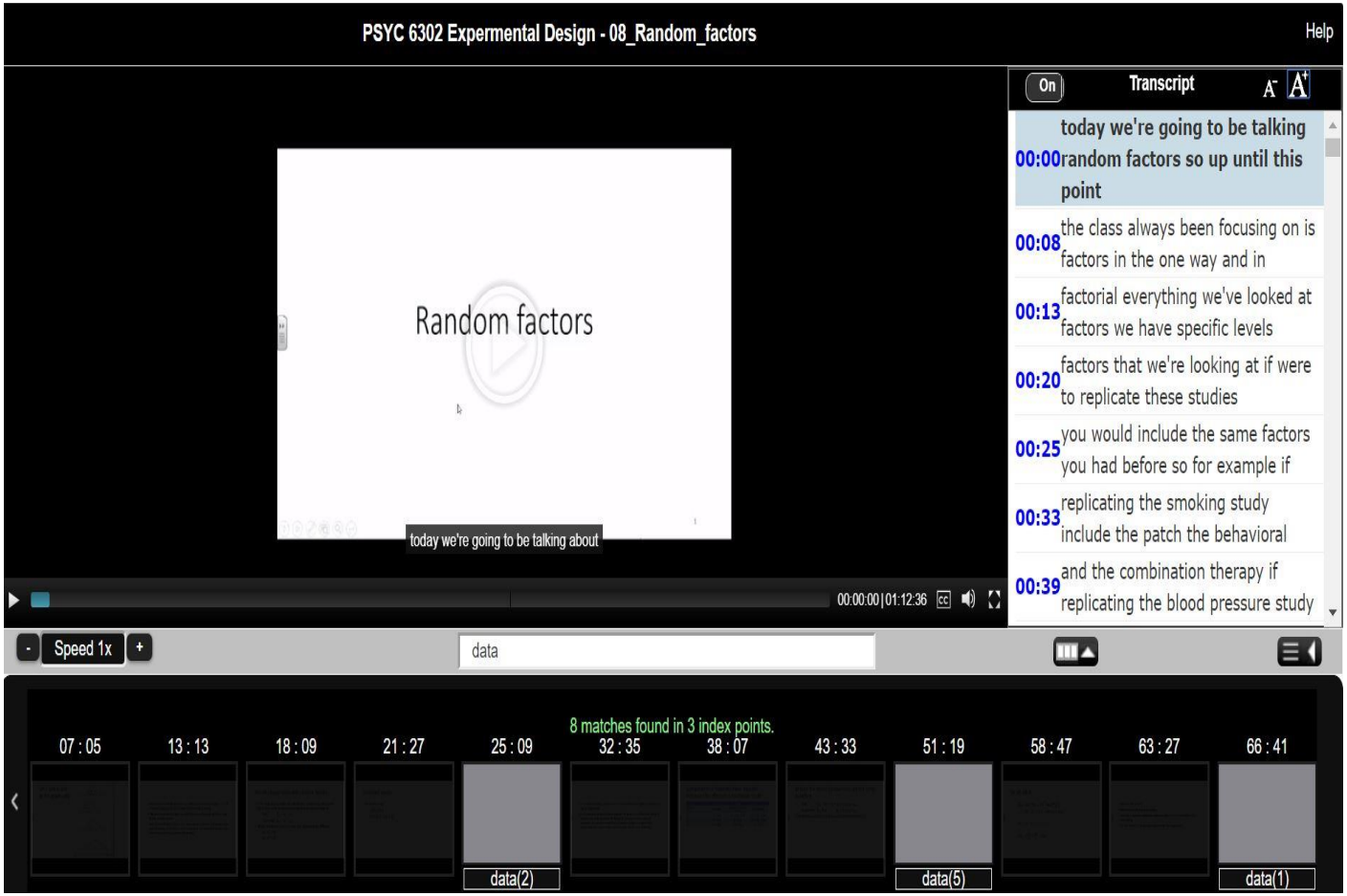

Figure 2.4 ICS Player with Search Functionality

When the user searches for a keyword from the search interface, the search module in the player activates and identifies a series of index points, allowing the user to navigate to the corresponding video segment.

Figure 2.4 represents the ICS player with the Search functionality enabled. In the example, the user has searched for a keyword ‘data,’ and the index panel shows the segments with the matching keyword. In this case, three index points have the word ‘data,’ and it has occurred eight times in total in the video. By clicking the specific index points, the user can view the lecture which contains the searched keyword.

CHAPTER – 3: RELATED WORK

3.1 Keyword Extraction

Several projects have addressed the extraction of keywords from documents. The most commonly used statistical measure to extract significant words in an unsupervised way is TF-IDF [6]. TF-IDF stands for *Term Frequency-Inverse Document Frequency*. It is intended to reflect the importance of the word in a document. The authors normalized the terms TF and IDF in different ways and produced different variants [6] [7]. There are other works which extract domain keywords from the online news articles [8]. Several news articles are collected and manually tagged with their domains. Along with the traditional TF-IDF, the authors used a new measure called the 'Word Common Possession Rate,' which is the ratio of the number of domain documents containing the word to the total number of domain documents. The authors proposed this new measure to compensate the limitations of TF-IDF in identifying keywords related to the specific domain when the total number of documents belonging to the domain is small compared to the total number of documents. The TF-IDF is multiplied with 'Word Common Possession Rate' to rank the keywords. They listed the evaluation of this work as their future work.

The research done by Lee et al., 2008 suggested a different technique to extract keywords for topic tracking; especially in news articles, using the 'Table Term Frequency' (TTF) in conjunction with the conventional TF-IDF [7]. As the first step, they compute TF-IDF for

all the words in each document and consider top n% of the words from each document for further processing. They did not state the selection of the value ‘n’ in their research paper. In the second step, they computed the term frequency on the words extracted in step 1 and determined the importance of words. To further improve the accuracy of extracted keywords, they performed ‘Cross-domain Comparision Filtering.’ The standard deviation of the words identified as keywords is computed taking their ranks in different domains of news articles like sports and politics. If the result of the standard deviation is below a certain threshold, they removed the word from the list of keywords. They believe that, if a word is ranked high in one domain and low in another domain, the word should be given importance in the domain which it is ranked high [7]. This work does not present any results.

Though a good amount of research has been done on extracting keywords, all of these techniques were applied either on documents or news articles which are well-phrased. In my scenario, I am utilizing the text extracted from video frames using OCR, and the output given by OCR does not follow the standard structure of the text documents. The importance of a word in a video also depends on several other factors like frequency, font size, the display time of the word on the frame. Owing to these limitations, directly applying the existing methods is not sufficient.

3.2 OCR Error Correction

The output generated by OCR may contain significant errors, and its quality depends on many factors like the quality of image fed to the OCR engine, combinations of text and background colors on the image, and use of small and exotic fonts [9]. The research work by Tuna et al., 2017 [9] applied several image enhancement techniques as pre-processing steps before the application of OCR. The image enhancement techniques employed in their work are 'Text Segmentation' and 'Color Inversion.' Text segmentation extracts the regions of text in an image by following a series of operations like binarizing an image using thresholding, dilation, edge detection using Sobel operator, blob coloring, and resizing.

Several post-processing techniques are widely applied to correct the errors during OCR text recognition [10] [11]. The work by Delden et al., 2004 [10] stated a supervised and an unsupervised approach for automatic spell correction. In the supervised approach, they focused on misspelled words which are mainly typing errors. They used a technique called the 'Reverse Minimum Edit Distance,' which takes the misspelled words and generates possible words using any of the four operations (Insert, Delete, Substitute, Transpose). Out of these words, they identified words part of lexicons (a pre-defined dictionary). If a word contains a multi-error, then the 'Reverse Minimum Edit Distance' technique is repeated for valid words obtained in step 1. This process continues until they get a reasonable candidate list. Then they map the misspelled words with the candidate words based on the similarity. In the unsupervised algorithm, they calculated the word frequencies in the

document and identified the words below a certain threshold as candidate errors. They applied specific rules on the candidate words using the list of common prefixes and suffixes to match them with the high-frequency words. These approaches resulted in an accuracy of about 75%.

Bhardwaj et al., 2008 [11] constructed a topic-based language model for every document in the training data and categorized the topics manually. MAP (Maximum A Posteriori) was used as an estimation instead of ML (Maximum Likelihood). For every topic, they created a language model using the Naive Bayes approach. 450 documents were used: 380 for training and 70 for testing. Their results show that there is an increase in accuracy of 25% compared to the standard approach.

All the methods stated above use some level of supervision that required human effort and time. It would be difficult to apply them to my problem as the domain of lecture videos is not limited. So, I have used an approach proposed by Youssef et al., 2012 [12] to correct the OCR errors using the Google Spell Correction API (Application Programming Interface), 'Did you mean,' as one of the steps in the auto-spell correction task. In this method, the text is divided into blocks of fixed size and sent to the API. If there is an error in the text submitted, the API suggests the best possible alternative using the probabilistic n-gram model.

CHAPTER – 4: AUTOMATIC KEYWORD EXTRACTION

This section details the algorithms and methodology used in the automatic extraction of keywords from lecture videos.

Keywords are words that would help to identify, at a glance, the nature of the material a segment of a video covers to improve navigation within the video lecture. A set of keywords summarize the content of a video segment. So, my main aim was to create a ranked list of words, based on their significance, for each segment by scrutinizing distinct attributes of text captured from the video frames.

OCR detects the text from video frames during the segmentation process of a lecture video and the keyword detection algorithm takes this text as input. To generate keywords, I have information about the text from the video frames and transcripts from the automatic speech recognizer. This thesis focuses on the text from the video frames with the assumption that the text from frames presents concise information on a topic. The assumption is that speech elaborates topics presented on frames.

While OCR technology has improved significantly over the years, the quality of the output depends on the source. In this scenario, I do not have access to the original lecture slides. These slides are available as a video that is converted to distinct images and fed to the OCR. In this process, some errors might be introduced in regenerating original screen text.

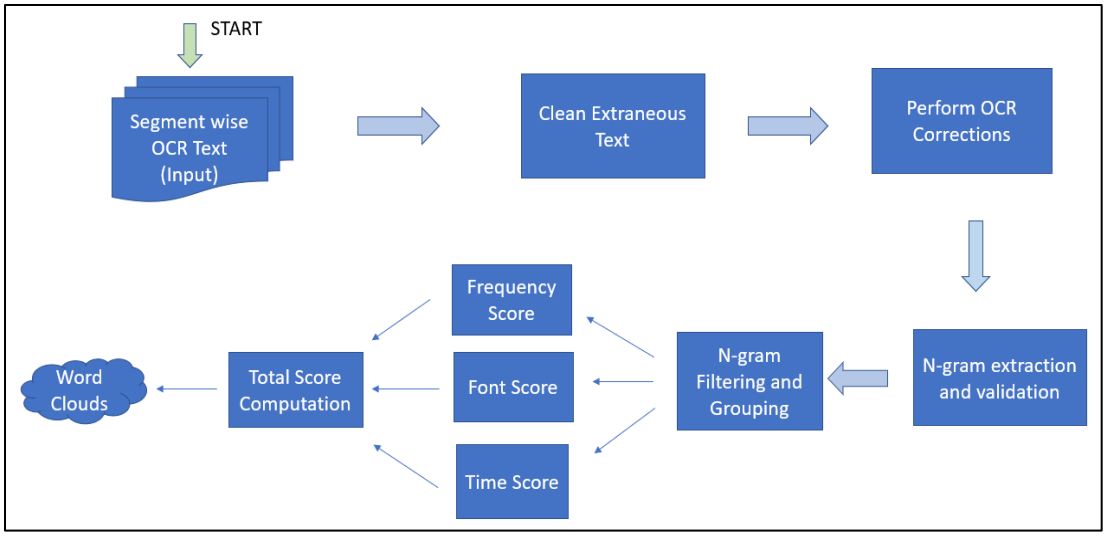

Figure 4.1 Steps in Keyword Extraction

Figure 4.1 represents the different steps involved in the process of keyword extraction. The process starts with segmented OCR text which is cleaned by removing extraneous text and applying auto-correction techniques. N-grams are extracted in the next step and validated to retain the meaningful ones. I then filtered the n-grams by eliminating stop words and grouped them using the stemming algorithm. I calculated different scores based on frequency, font, and time and combined them to compute the final score for each n-gram. I elaborated the details of each step in later sections.

4.1 Discard Extraneous Text

Depending on the style the instructor chooses to capture their lecture, the recorded video may include irrelevant text from the computer. For example, the taskbar can be captured and may be present in a part of the lecture or an entire lecture. There are some words which

could occur on all or most of the slides of the presentation. Sometimes, the instructor uses the words like university name, or the instructor's name, or the course name on each slide of the lecture and these words appear almost at the same positions on all the slides. These words do not represent the content discussed in the segment. I want to ensure these words are not identified as keywords.

To handle these scenarios, I came up with an algorithm to identify words repeating at similar positions across the lecture. OCR gives positional information of every word like Left, Right, Top, Bottom. Using this position as a reference, the algorithm keeps track of how many times a word has occurred in a specific position on every transition point. If a word has occurred more than a certain threshold at a particular position, the algorithm eliminates the word from further processing. I have set 40% of the total transition points present in the lecture as a threshold by observing a few lectures.

The reason for limiting the word elimination to a specific position is to retain the word that has occurred at other positions holding some significance. For instance, the name of the course is ‘Computer Architecture,’ and assume it has occurred on all the slides of the course. There is another word ‘Fourth Gen Computer’ occurring somewhere on one of the slides. With the approach used, the algorithm considers the words ‘Computer’ and ‘Architecture’ separately for elimination as the process deals with unigrams. Now, discarding the word 'Computer' at all positions will also eliminate the word 'Computer' in the word sequence 'Fourth Gen Computer'. To avoid this information loss, I designed the algorithm to eliminate the words based on positions.

There is a justification for using the phrase ‘similar positions’ instead of ‘same positions.’ There is a chance that the words occurring across the lecture may slightly change in position relative to another slide. I have allowed a 5 px deviation by measuring the Euclidean distance between the position of words in two different slides.

As the OCR is not perfect, it may mispredict the letters in words. It may recognize the letter ‘i’ as the letter ‘l,’ the number ‘0’ as the letter ‘o,’ and so on. To handle this, I identified words in similar locations even in the presence of small OCR errors by computing *Cosine Similarity*. Cosine similarity is the cosine of the angle between two vectors [13]. It represents text as vectors and calculated as the dot product of two vectors to the cross product of the two vectors. It ranges from 0-1. If the similarity between the two words is greater than 0.5, I considered the words as the same and grouped them. The following algorithm presents the process of cleaning the extraneous text.

Function Name: *cleanExtraneousText*

Input: List of words recognized by OCR with positions

Output: List of words after removing irrelevant words

Track all the positions of each word and maintain the count of no. of times of occurrence at the same position

for each word in words:

Group all the positions which are less than 5 px distance and add their occurrences

if occurrences > threshold then

Add to elimination list

Using position as key, find all words occurring in that position

Now compare the cosine similarity scores of words occurring in each position

if similarity > 0.5 then

Add the counts of 2 words

if count > threshold then

Add to elimination list

Remove words present in elimination list with respect to their position

The following algorithm details the process of grouping the words based on the word positions.

Function Name: *groupSimilarPositions*

Input: Word with positions and count sorted in increasing order of the count

Output: Updated count of occurrences of word after grouping similar positions

```
positions = list of all positions and their respective counts
nearestPositionList = position having highest occurrences
for each position in positions:
    for each nearestPosition in nearestPositionList:
        if distance between nearestPosition and position < 5 then
            Add to position to nearestPositionList
            Add the occurrences
            break
```

4.2 OCR Error Correction

The results given by OCR are imperfect as it makes mistakes in recognizing characters. Due to this, relying entirely on the text identified by OCR does not guarantee a good set of keywords. So, I performed some post-processing steps on the output given by OCR to make possible improvements to the text. There are many approaches which use the supervised model by maintaining a set of lexicons [10] or the statistical model [29] using the Naive Bayes. All these approaches demand huge dictionaries covering all the terms of the domain to get accurate results. These techniques may not fit the field of education as it does not restrict to one domain and the knowledge base keeps updating constantly. Manually rejuvenating the data requires a lot of effort. The approach I have implemented uses the method suggested by Youssef et al., 2012 [12] to correct errors. They have proposed a context-based error correction technique using Google's Spell Suggestion API.

There are two types of errors generated by OCR [14]: Non-word errors and Real-word errors. Non-word errors occur when the words detected by OCR are not part of the dictionary. For example, OCR recognizes the word 'probability' as 'jrolability.' Maintaining a dictionary could solve these issues; however, the dictionary should also enclose the names of persons, and locations [12]. Real-word errors occur when the dictionary encompasses the words detected by OCR. However, they don't fit the context. For instance, OCR recognizes the n-gram 'state transition probability' as 'stale transition probability'. To solve the linguistic and context-based errors, I used an API (Application Programming Interface) provided by Google known as 'Did you mean'. Google has indexed several

trillion web pages and contains an enormous collection of words and n-gram sequences that could best serve any application related to speech and text recognition. Using this plethora of information, I developed a method to correct OCR errors.

Firstly, all the text from each transition point was taken and divided into blocks of fixed size [12]. Every block contained a fixed number of words. The Google Search API takes every block as a query and makes suggestions. The idea behind sending a block of text rather than a single word is the assumption that the prediction is accurate when the context surrounds the word. The results returned by the search API were parsed to see if it contains the phrase 'Did you mean.' If the phrase was present, it indicated the existence of a misspelled word, and the algorithm replaced the old block of text with the new result. This process continued until all the text from the video completed its validation. Figure 4.2 demonstrates the 'Did you mean' feature. Google suggests the best alternative for the search made with incorrect spelling.

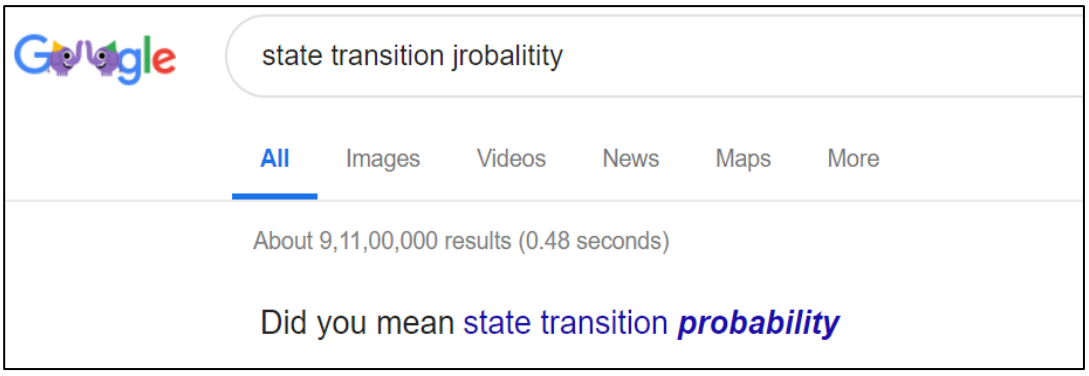

Figure 4.2 Demo of 'Did you mean' Feature

I experimented with different sizes of blocks, and the results were optimal with block sizes of 10 to 20 words. Figure 4.3 shows the text before and after the spell correction with ten words per block.

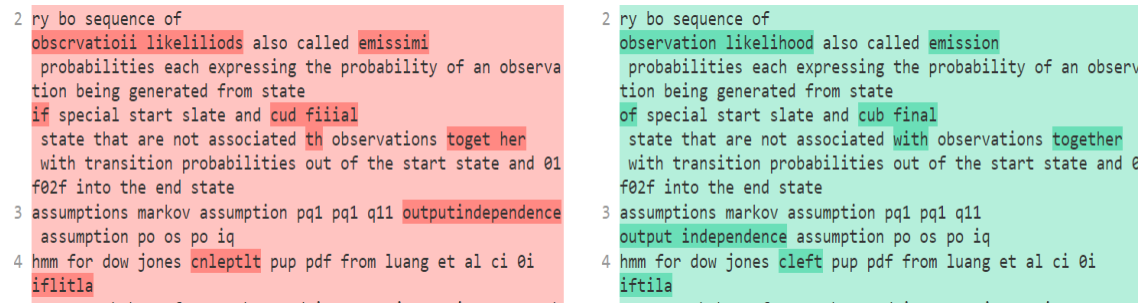

Figure 4.3 Text Before and After Spell Correction

The text to the left represents the original text from OCR and text to the right is the text after performing the correction. Figure 4.3 also highlights the words which differ before and after the spell correction. This process corrects some of the important words like ‘observation likelihood’ and ‘emission’. It also modifies some words incorrectly; for example, the word 'cud' changes to 'cub,' which should be 'end' in this context. Also, there are some words which are unmodified even though they are not meaningful words.

Since OCR extracts the text from the image left to right, top to bottom there is no notion of sentence boundaries. The formatting style of the slide varies from instructor to instructor. Some instructors may use up the entire width of the slide, and some use it by splitting into two or more columns and so on. With these limitations in place, I cannot say that the output from the OCR has its full context preserved. This could be a reason why the Google API misinterpreted some of the words.

To improve the correction further, I developed a method to perform another round of spell correction with the corrected text as input. In this approach, the algorithm scans through the text corrected by Google and identifies valid and invalid words. For a word to be valid, it should occur at least three times, and it should be composed of alphabets (A–Z). The reason for using the count as a criterion is, the probability that the OCR detects one word as another incorrectly, in the same way, is very low. For example, suppose OCR recognizes the word 'state' as 'stale' at one instance, there is only a miniscule chance that OCR detects the word as 'stale' multiple times. Also, the first phase of spell correction could help to correct some of these errors. The algorithm considers all the other words as invalid. Using the list of valid words as lexicons, the algorithm suggests spell corrections for invalid words utilizing the concept of *Edit Distance* [15]. Edit distance is the minimum number of operations required to convert one string to another. If the edit distance between the invalid word and suggested word is less than five, the suggested word replaces the invalid word. The value five is chosen by experimenting with words from different lectures. Figure 4.4 shows a demo of spell corrections using the above process and the algorithm in the following page details the process.

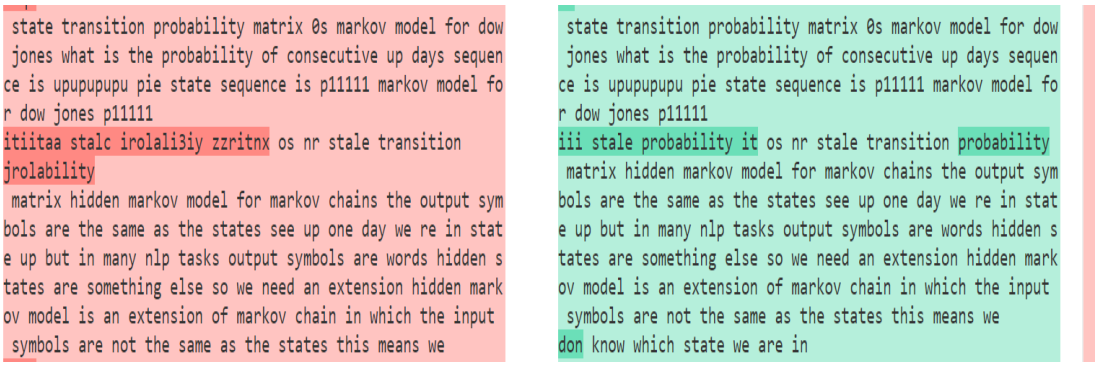

Figure 4.4 Demo of Corrections in Phase 2

From Figure 4.4, I noticed a slight improvement in the text. Even after performing two rounds of spell correction, I still found the existence of noise in the text. Some arise due to the presence of formulae and algorithms on the slide where OCR does a poor job in predicting them. The current methodologies cannot correct these errors. Sometimes, OCR combined two adjacent words, missing a few characters in each word. The current algorithm detected at least one word correctly most of the times. For example, 'rcturnsfiriiardprob' is predicted by OCR as a single word, which is a combination of two or more words. The API suggests the word as 'return,' correcting at least one word.

Function Name: *refineOCRCorrections*

Input: Corrected OCR text from Google API

Output: Text with enhanced corrections

```
    Identify valid and invalid words from the input
    for each word in invalid words:
        Get suggestions using valid words as a dictionary
        Find edit distance between invalid word and suggested word
        if distance < 5 then
            Replace invalid word with suggested word
```

4.3 N-gram Extraction and Validation

The OCR text was processed to remove most of the extraneous text and make viable corrections on the incorrectly identified words. The next step is to extract the n-grams from the processed text. N-gram is a contiguous sequence of 'n' terms from the given sequence of text [16]. In this research, I extracted unigrams, bigrams, and trigrams as candidates to be keywords. However, the code has been designed to easily extend to any n-grams, with changes to the configuration file.

OCR scans an image left to right, top to bottom and does not guarantee to maintain contextual relation between the words. Punctuations would help identify sentence boundaries, but OCR does not capture this information. These limitations result in generating extra n-grams: for example, a bigram formed using the last word of one sentence and the first word of another sentence.

Collecting relevant domain-specific books and constructing a statistical language model can solve the issue of handling extra n-grams. The statistical language model gives probabilistic distribution over the word sequence [17]. Markov's assumption simplifies the calculation of n-gram probability. Given a sequence 'There was heavy snowfall,' the steps to calculate the probability of the n-gram are as follows:

P('There was heavy snowfall') = P('There,' 'was,' 'heavy,' 'snowfall') ---------- (1)

P('There was heavy snowfall') = P('There') P('was' | 'There') P('heavy' | 'There was') P('snowfall' | 'There was heavy') ---------- (2)

Using Markov's assumption, equation (2) is simplifies to

P('There was heavy snowfall') = P('There') P('was' | 'There') P('heavy' | 'was') P('snowfall' | 'heavy')

This method ranks the n-gram sequences using the probabilities and ignores the sequences with lower values.

This process could be a good starting point to validate the n-grams. However, this process would require manual effort for collecting and organizing the digitized books related to the lecture's domain. Instructors use information from diverse sources to prepare the lecture content and gathering all the sources of information would be a tedious task. I addressed the problem of manual book collection by using an API called 'PhraseFinder' [18].

Google provides a service called 'Google Books Service' which aims to allow people to search for the content in books. Google has scanned over five million books published between the year 1500 – 2009 and generated a large corpus of words to get an estimate of the word usage with its context [19]. It contains around eight billion phrases (unigrams to 5-grams). This information is beneficial for validating the n-grams. PhraseFinder, a search engine for Google n-gram dataset, acts as an interface to provide the statistical data of n-grams, and I utilized this service in my research.

Figure 4.5 and 4.6 give two examples using the PhraseFinder API. The API takes the phrase as input and provides the count of occurrences, and the number of books in which it has occurred including the range of years of the books. It lists the phrase separately based on

the case (lower case, upper case, and camel case). From the following examples, the phrase ‘programming language’ is more frequently used than the phrase ‘programming note’.

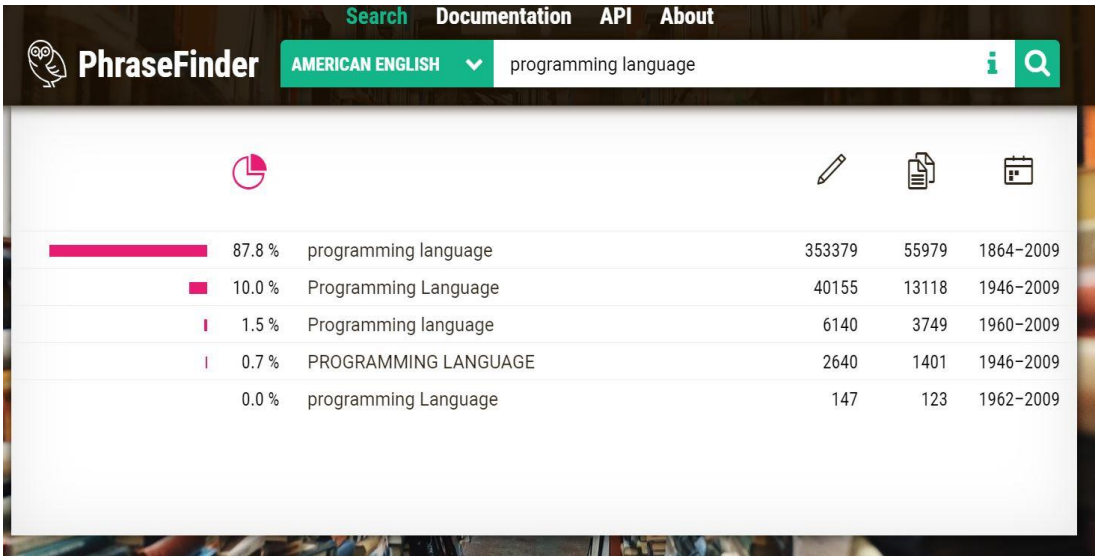

Figure 4.5 Example-1 of PhraseFinder

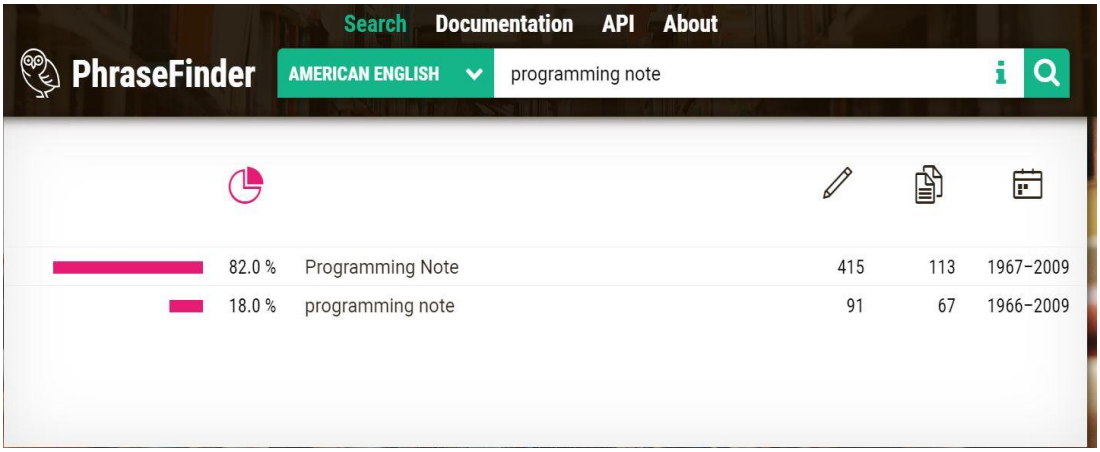

Figure 4.6 Example-2 of PhraseFinder

Considering the results given by this API, the keyword detection algorithm validated the n-grams by discarding the ones with count 0. The algorithm maintained two counts for all the n-grams: local count and global count. Local count gave the information of how many

times a phrase has occurred in a segment, and global count was the count provided by the PhraseFinder API. For computing the n-gram score based on frequency, the keyword detection algorithm utilizes the local count. Discarding the n-grams with count 0 would aid to clear the invalid phrases and those which contain spelling mistakes. The keyword detection algorithm also eliminated n-grams containing repeated words (e.g., 'data programming data' was eliminated due to the repetition of the word 'data') and which start or end with stop words, as keywords generally do not contain them in those positions.

The algorithm used the information from external sources as the content on the slide is very sparse. Some lectures contain very few words and considering the frequencies only from the lecture can be misleading. Also, the information from the external sources helps to validate a phrase over a massive database of books. This process generated a list of n-grams for each segment of the video.

4.4 N-gram Filtering and Grouping

In this step, I filtered and grouped the n-grams for each segment with different parameters and this section details them. All the words are case-folded by converting to lower case.

4.4.1 Stop word Removal

'Stop words' are frequently occurring and trivial words which help frame sentences but do not represent the topics discussed. Articles, prepositions, conjunctions, and pronouns are

typically stop words. Examples of stop words include ‘a,’ ‘an,’ ‘the,’ ‘it,’ ‘and,’ ‘as,’ ‘what,’ ‘how’. I collected the list of stop words from an external source [20].

The keyword detection algorithm can take either TRUE or FALSE for the parameter *stopwordRemoval*. If the parameter is TRUE, it eliminates the n-grams containing stop words. If the parameter is FALSE, it retains the n-grams with very common stop words like ‘and,’ ‘to,’ ‘of,’ ‘of,’ ‘on,’ ‘by’ and eliminate the n-grams containing rest of the stop words. Most of the valid n-grams contain the abovementioned stop words, and the chances of valid n-grams containing the other stop words are very less.

4.4.2 Stemming

Stemming is the process of reducing words to their roots or *stems* [21]. A stem or root is the part of word retained after removing its suffix. This process groups all the words in different forms to one stem. For instance, nouns have plural forms (e.g., ‘Computer’ and ‘Computers’), and verbs have gerund form (‘ing’ as the suffix), and verbs in present tense differ from past tense.

I analyzed the output of Porter stemmer, Lancaster stemmer, and Snowball stemmer. Porter’s algorithm [22] is a rule-based stemmer which has a pre-defined set of rules to stem the words. It can produce stems which may not be readable words. Lancaster is an iterative stemming algorithm defined over 120 rules [23]. It stems very aggressively, sometimes leading to faults. Snowball [24], which is also known as Porter2, is the enhanced version of Porter’s algorithm with more sophisticated rules. The Snowball stemmer also produces

stems. I have used the Snowball stemming algorithm in my thesis. Since the Snowball stemming algorithm produced stems, I replaced the stems with words to make them presentable to the user.

Function Name: *decideWordToStem*

Input: Stemmed word

Output: The most common word for the stem

Find the list of all words mapping to the same root

Out of all the words for a single root, check the word which occurred max no. of times and select the word.

The above algorithm elaborates the process of replacing stems with a valid word. For example, suppose the words ‘dissolved’ and ‘dissolving’ have been reduced to ‘dissolv’ by the stemmer, the algorithm replaces the stem either by ‘dissolved’ or ‘dissolving’ based on their frequency of occurrence. If ‘dissolving’ has occurred five times and ‘dissolved’ has occurred two times, then ‘dissolving’ is given as output. If both the word frequencies are the same, then without loss of generality, the algorithm selects the word with more characters (letters). The algorithm groups the words based on their stems and updates the local count of the selected word. From the previous example, the selected word ‘dissolving’ will have the weight updated to 7 (5+2).

In the case of n-grams, the algorithm reduces each term of n-gram and decides the word to be displayed. Consider the n-gram ‘computer program’ occurring four times and ‘computer programs’ occurring five times. The stemming process happens as follows:

Stemming Process

Step 1: Reduce the n-grams by considering each term

‘computer program’ reduced to (comput, program)

‘computer programs’ reduced to (comput, program)

Stem 2: Group the words having the same stems and update the count

(comput, program) → [computer program, computer programs] → the count is 9 (4+5)

Step 3: Based on the criteria set by the algorithm, select the word to display using the function *decideWordToStem* and update the weight.

In this case, the word selected was ‘computer programs’ as it has a higher frequency. The new weight of the word is 9.

The reason to break the n-gram to separate terms is that the stemming algorithm only focuses on the suffixes of the last word rather than stemming all the words. To get accurate results, the algorithm splits the n-gram to ‘n’ terms.

The keyword detection algorithm can take either TRUE or FALSE for the parameter *applyStemming*. If the parameter is TRUE, it reduces the n-grams and updates the weights accordingly. If the parameter is FALSE, it does not make any modifications to the n-grams.

The process of stemming is applied to each segment independently. If the word ‘computer program’ occurred in one segment and ‘computer programs’ occurred in another segment, the algorithm does not group them.

4.5 Frequency Score

Frequency is an important attribute that determines the importance of an n-gram. The frequency score of an n-gram internally depends on a few measures as stated below:

- **Term Frequency:** This measure determines the count of an n-gram in a segment.
- **Inverse Segment Frequency:** This measure determines the relative importance of an n-gram in a segment.
- **Domain Importance:** This measure identifies n-grams containing domain related words and assigns higher importance.
- **Reduce Weights of Rare Words:** This measure identifies n-grams containing rare words and reduces their weight.

These measures are elaborated in the following sections.

4.5.1 Term Frequency

Term Frequency (TF) is the number of times an n-gram occurred in a segment. It represents the weight of the term in a segment. This statistical measure is beneficial in unsupervised identification of keywords. This measure is mandatory to obtain the n-gram weights.

TF (n-gram) = count of n-gram occurrences in a segment

The method I developed takes care of the counts of lower order n-grams that are already part of the higher order n-grams. Consider the example weights of the n-grams as follows:

Table 4.1 N-gram Original Weights

N-gram	Count
software development practices	5
software development	7
development	10

In the example demonstrated, the word ‘software development practices’ occurs five times, ‘software development’ occurs seven times and ‘development’ occurs ten times. When the algorithm calculates the count of the unigram ‘development,’ it also counts its occurrences from the bigram ‘software development’ and trigram ‘software development practices’. This does not represent the exact weight of the unigram. To handle this effectively, the keyword detection algorithm subtracts the frequency of higher order n-gram from the frequency of lower order n-gram, if both the n-grams have some terms in common.

The weight of ‘software development’ which is seven becomes 2 (7 – 5), as the word is already part of the trigram five times. Similarly, the weight of the word ‘development’ becomes 3 (10 – 5 – 2). Ten is the total number of times the word has occurred as a unigram, five is the number of times the word has contributed to the trigram, and two is the contribution for the bigram. Table 4.2 shows the updated weights of the n-grams.

Table 4.2 N-gram Updated Weights

N-gram	Count
software development practices	5
software development	2
development	3

4.5.2 Inverse Segment Frequency

A measure which tells how much information a word can provide is Inverse Document Frequency (IDF). It works on the idea that if a word is too frequent in most of the documents, then it may not constitute any significance and may not be the best keyword. So, IDF analysis tries to reduce the weights of such words giving scope for other words to become keywords. In this thesis, I considered IDF as ISF (Inverse Segment Frequency). This measure tries to identify the words that are very frequent across different segments of the video and reduce their weights.

$$IDF \text{ or } ISF \text{ (n-gram)} = log \frac{Total\ no.of\ segments\ in\ the\ video}{Total\ no.of\ segments\ a\ word\ has\ occurred + 1}$$

The addition of '+1' in the denominator avoids division by zero error.

Consider a video with 13 segments. A word 'w_1' has occurred in 12 segments out of 13, and another word 'w_2' has occurred in three segments. Computing the ISF results in the following scores.

$ISF(w_1) = \log \frac{13}{12+1}$

$ISF(w_1) = \log \frac{13}{13}$

$ISF(w_1) = \log 1$

$ISF(w_1) = 0$

Similarly, ISF for the word w_2

$ISF(w_2) = \log \frac{13}{3+1}$

$ISF(w_2) = \log \frac{13}{4}$

$ISF(w_2) = \log 3.25$

$ISF(w_2) = 0.511$

From the above results, I can say that $ISF(w_2) > ISF(w_1)$

The keyword detection algorithm can take either TRUE or FALSE for the parameter *applyISF*. If the parameter is TRUE, the algorithm calculates the weight of the n-gram as the product of TF and IDF (ISF). If the parameter is FALSE, the weight of the n-gram is equal to the raw count of the term in the segment, i.e. TF. So, the weight of the word is either TF or the product of TF and IDF depending on the value of the parameter provided.

4.5.3 Domain Importance

Another crucial factor considered to boost the weights of the keywords is domain importance. The basic ideology behind this is to assign more importance to the words which are directly related to the domain of the lecture. For example, for a lecture which belongs to Chemistry domain, the keyword detection algorithm will give more importance to the words from the Chemistry domain rather than general words. If two words ‘solution’ and ‘application’ are present in a segment of a video lecture, the word ‘solution’ is assigned more weight than the word ‘application.’ In this way, the keyword detection algorithm does not miss any relevant words in the process of keyword extraction.

The domain-specific words are identified using information from external sources. I have collected the data provided by Oxford Reference Dictionaries for different domains [25]. Oxford dictionaries are considered one of the standard sources of information and widely accepted by many authorities. Hence, this is used as a look up to decide if the word has a reliance on the domain.

When the parameter *assignDomainImportance* takes the value TRUE, the keyword detection algorithm multiplies the weight by a specific factor when the word is related to the domain. If the parameter is FALSE, the algorithm does not amplify the weight.

With n-grams, the keyword detection algorithm splits them into ‘n’ terms and checks for each term if it is related to the domain and multiply the weights generated by step 4.5.2. This gives the enhanced weight of the n-gram.

The process starts with assuming a factor for increasing the words which are domain related. In this research, I have used a factor 2^k, where 'k' is the number of domain-specific words in an n-gram. The 'boostedWt' is set to 1 initially for every n-gram.

Function Name: *assignDomainImportance*

Input: n-grams with their weights and domain to which the video belongs

Output: n-grams with modified weights

```
Load the list of domain-specific words
domainWtFactor = any constant value greater than 1
for each n-gram in n-grams:
    Split n-gram to 'n' terms
    Assign boostedWt = 1
    for each term in n-gram:
        if term in domain-specific words:
            boostedWt = boostedWt * domainWtFactor
    ngramWt = ngramOriginalWt * boostedWt
```

Subsequently, the weights are updated as explained in the algorithm. This process was repeated for all the n-grams in each segment of the lecture.

This step ensures that the n-grams with more domain-specific words are more likely to make to the list of keywords extracted for that segment.

4.5.4 Reduce Weights of Rare Words

The next factor considered is reducing the weights of rare words to reduce their chance of making into top keywords. Rare words are words which occur occasionally and do not belong to either the domain-specific category or the general English category. These words can be OCR errors or some abbreviated terms which are specific to the instructor or the class. Most of the OCR errors are either corrected by OCR error correction (Section 4.2) or eliminated during the validation of n-grams using the PhraseFinder API (Section 4.3). However, some erroneous words are not eliminated. For example, when the OCR detects a word incorrectly many times and makes the word valid when performing the second round of spell correction or the Google n-gram data set contains the word since Google also utilizes OCR technology to scan the books.

This process closely follows the method discussed for domain importance. The algorithm divides the n-gram into ‘n’ terms and reduces each term’s weight by dividing it with a certain factor. In this work, I have assumed the factor as 2^k, where ‘k’ is the number of rare words present in an n-gram. The algorithm takes the n-gram weights and the domain of the video as input. Oxford dictionaries acted as a source for domain-specific words, and python libraries like PyEnchant and NLTK Wordnet served a reference for English words.

The algorithm checks every term in the n-gram and reduces its weight if it contains rare words.

This parameter can also be set to TRUE or FALSE. If *reduceRareWordWeights* parameter is TRUE, the keyword detection algorithm modifies the weights. If it is FALSE, the algorithm does not alter the weights.

Function Name: *reduceRareWordWeights*

Input: n-grams with their weights and domain to which the video belongs

Output: n-grams with modified weights

```
Load the list of domain-specific words
rareWtFactor = any constant value between 0 and 1
for n-gram in n-grams:
    Split n-gram to 'n' terms
    Assign reducedWt = 1
    for each term in n-gram:
        if term not in domain-specific words and not a valid English word:
            reducedWt = boostedWt * rareWtFactor
    ngramWt = ngramOriginalWt * reducedWt
```

4.5.5 Final Frequency Score

The current system to extract keywords performs the steps discussed in Section 4.5.1 – 4.5.4 sequentially and determines the weights of the n-grams for each segment in the video lecture. These weights can range from zero to a higher value. To standardize this, the keyword detection algorithm normalizes all the weights using 'Min-Max Normalization.'

Normalization is a mapping technique which maps the existing range of values to a new range [26]. It linear transforms the values from the original data.

$$New\ value = \frac{Old\ value - Min\ value}{Max\ value - Min\ value} \ .$$

To explain it in this scenario,

$$Final\ Frequency\ Score = \frac{Frequency\ Score - Min\ Frequency\ score\ of\ segment}{Max\ Frequency\ score\ of\ segment - Min\ Freqency\ Score\ of\ segment} \ .$$

The keyword detection algorithm takes the minimum and maximum values for each segment and determines the new values. This step gives the final score of the n-grams based on the frequency.

4.6 Font Score

The second attribute considered in my thesis is the font size of the word present on the video frame. OCR gives information about the font size of each word, and my algorithm utilized this information to determine the weight of the n-gram based on font size. This turns out to be another way to investigate the problem of determining keywords. The words

which have larger font sizes are more important than the words with relatively smaller fonts.

4.6.1 Determining Font Weights of N-grams

Firstly, all the font sizes of the words are squared. To distinguish the difference clearly, the algorithm takes the square of the values. For example, assume the word 'w_1' has a font size of 13, and the word 'w_2' has a font size of 14. The difference between font sizes of both the words is 1, and it becomes negligible when the scores are normalized. To differentiate the importance of words, the algorithm squares the font weights, and they become 169 and 196, respectively. Thus, providing a reasonable margin to distinguish the significance clearly.

The same word can occur at multiple places with different font sizes in a segment. So, the algorithm estimates the average font weights of the word across each segment of the video. Let us consider the following table with three different words and their weights occurring at various places in a segment.

Table 4.3 Snippet of Words with their Font Size

Word	Original Font Weight	Squared Font Weight
$\mathbf{W_1}$	10	100
$\mathbf{W_2}$	10	100
$\mathbf{W_1}$	13	169
$\mathbf{W_3}$	10	100
$\mathbf{W_1}$	7	49
$\mathbf{W_3}$	7	49

$$\textit{Average weight of the word} = \frac{total\ weight\ of\ the\ word}{no.\ of\ occurrences\ of\ the\ word}\ .$$

Weight of $W_1 = \frac{100 + 169 + 49}{3} = \frac{318}{3} = 106$

Weight of $W_2 = \frac{100}{1} = 100$

Weight of $W_3 = \frac{100 + 49}{2} = \frac{149}{2} = 74.5$

Table 4.4 Snippet of Words with their Average Font Size

Word	Average Font Weight
$\mathbf{W_1}$	106
$\mathbf{W_2}$	100
$\mathbf{W_3}$	74.5

The keyword detection algorithm calculates the weights for each segment separately, i.e. if the word 'W_1' has occurred in another segment, it does not include that in the calculation of its weight in the current segment. The following algorithm explains the process.

Function Name: *calculateFontWts*

Input: Words with font size for a segment of the video lecture

Output: N-gram weights using font weights

```
Square the font weights of every word and calculate the average font weight of
the word

Get the n-grams for the specific segment of the video lecture

for each n-gram in n-grams:

        Split the n-gram to 'n' terms

        ngram_wt = 0

        for each term in terms:

                ngram_wt = ngram_wt + average weight of the term

        ngram_wt = ngram / no. of terms
```

Once the keyword detection algorithm determines the average weights of the words in each segment, then it takes the n-grams obtained from Section 4.4 and determines the score

based on font. The process proceeds by splitting the n-gram into 'n' terms and taking the ratio of the sum of the average weights of the terms to the total number of terms.

$$\textit{Font-score of n-gram} = \frac{sum\ of\ average\ weights\ of\ 'n'\ terms}{n}$$

For instance, to calculate the weight of the bigram 'W_1 W_3', the algorithm splits the bigram into two terms 'W_1' and 'W_3'. Using the average weights calculated in Table 4.4, it computes the font score of the bigram as follows.

$$\textit{Font score of '}W_1\ W_3\textit{'} = \frac{Average\ weight\ of\ W_1 + Average\ weight\ of\ W_3}{2}$$

Font score of 'W_1 W_3' = $\frac{106 + 74.5}{2}$ = 90.25

4.6.2 Final Font Score

Similar to the frequency score, the keyword detection algorithm normalizes the font score to range between 0 and 1. Min-Max normalization is applied to the original weights to get the final score of the n-grams based on the font size.

$$\textit{Final Font Score} = \frac{Font\ Score - Min\ font\ score\ of\ the\ segment}{Max\ font\ score\ of\ the\ segment - Min\ font\ score\ of\ the\ segment}\ .$$

4.7 Time Score

Another essential attribute considered to determine the importance of an n-gram is its time score. The keyword detection algorithm tracks the time each video frame is displayed and determines the time each n-gram is present on the segment. Longer the time the n-gram is present, the higher is its importance. The interpretation is that, when an instructor emphasizes a specific slide for more time, it may be important. Some slides appear for a short period of time like some announcements, giving a brief overview of the previous class, and text present on these slides is relatively less important.

4.7.1 Determining Time of N-grams

The algorithm keeps track of the text present on each slide along with its duration and then groups the slides based on the segments. It sorts the n-grams obtained from Section 4.4 in the descending order of the value 'n,' i.e. all the trigrams will occur first, followed by bigrams and unigrams. An example is stated below to explain the reason for sorting the n-grams.

Table 4.5 Snapshot of Slide Content of a Segment

Slide no.	Slide Content	Time (sec)
1	W_1 W_2 W_3 W_4	60
2	W_1 W_2 W_3 W_2 W_3 W_5	20
3	W_2 W_3 W_4	30

The content in Table 4.5 tells that a specific segment of the video has three distinct frames or slides. It shows the text present on each frame and its duration. Let the valid n-grams for the segment be 'W_2 W_3', 'W_4', 'W_1 W_2 W_3'.

Now, the algorithm sorts the n-grams based on the number of terms each n-gram contains.

Table 4.6 Ordered N-grams for a Segment

Order	N-grams
1	'W_1 W_2 W_3'
2	'W_2 W_3'
3	'W_4'

To find the duration of each n-gram, it goes through every slide and adds the duration for which they occur. Once it counts the n-gram duration, the keyword detection algorithm temporarily discards the n-gram from the slide. This is to prevent adding the duration of lower order n-grams multiple times.

Flow

I list the steps in the example below:

1. Start with n-gram 'W_1 W_2 W_3' and scan through the three slides to get the duration. This n-gram occurs in slides 1 and 2 for 60 seconds and 20 seconds, respectively. The total time for the n-gram would be 80 seconds (60 + 20). Now remove the n-gram occurrences from the text. The new text on the slide becomes as follows.

Table 4.7 Slide Content after Step 1

Slide no.	Slide Content	Time (sec)
1	W_4	60
2	W_2 W_3 W_5	20
3	W_2 W_3 W_4	30

2. Take the next n-gram 'W_2 W_3'. Scanning Table 4.7, the n-gram occurs in slide 2 and slide 3 for a total duration of 50 seconds (20 + 30). Had the algorithm not discarded the higher order n-gram from the slide content and scan Table 4.6, the n-gram 'W_2 W_3' would have occurred in slides 1, 2 and 3 contributing to a total duration of 110 seconds (60 + 20 + 30) which is not a valid duration. This is the reason why I developed the algorithm to sort the n-grams based on the value of 'n' and remove them from the slide temporarily after computing their duration.
3. Processing the duration of n-gram 'W_4' gives 90 seconds (60 + 30).

Table 4.8 Slide Content after Step 2

Slide no.	Slide Content	Time (sec)
1	W_4	60
2	W_5	20
3	W_4	30

4.7.2 Final Time Score

To compute the final time score, the algorithm normalizes the time scores of the n-gram using 'Min-Max' normalization to range between 0 and 1.

$$Final\ Time\ Score = \frac{Time\ Score - Min\ time\ score\ of\ the\ segment}{Max\ time\ score\ of\ the\ segment - Min\ time\ score\ of\ the\ segment}$$

This gives the final score considering time.

The following algorithm shows the computation of time scores.

Function Name: *calculateTimeWts*

Input: All the text present on every slide of segment with its duration

Output: N-gram weights with importance of time

```
Get all the n-grams for the segment and sort them based on the value of 'n'
for each n-gram in n-grams:
    ngram_duration = 0
    Scan through all the slides and add the duration if n-gram exists on the slides
    for each slide in slides:
        if n-gram is present on slide:
            Update the n-gram duration
            ngram_duration = ngram_duration + slide_duration
            Remove the n-gram from the slide
```

4.8 Final Score Computation

The next step is to combine three different scores from three attributes: frequency, font and time. I introduced three new parameters called frequency weight, font weight, and time weight, whose values are between 0 and 1, and their combined total sums to 1. These new parameters were introduced to assign importance to frequency, font and time scores.

$$0 <= \text{frequency weight} <= 1$$

$$0 <= \text{font weight} <= 1$$

$$0 <= \text{time weight} <= 1$$

$$\text{frequency weight} + \text{font weight} + \text{time weight} = 1$$

Collecting the ground truth and performing evaluations can help determine the best values for these weights.

Final Score of N-gram *= (frequency weight * Final Frequency Score) + (font weight * Final Font Score) + (time weight * Final Time Score)*

4.9 User Presentation

Once the algorithm determines the final score of the n-grams, the top words for each segment are selected and shown as a summary to the user in the form of word clouds. The number of words to be displayed is determined by selecting the range for each segment.

Range = Maximum Final Score obtained in a segment – Minimum Final Score obtained in a segment

The algorithm selected the words which were above 40% of the computed range as keywords and presented them to the user. However, the maximum number of keywords shown in each segment was limited to 20. The selected n-grams were displayed to the user in the order of their importance by embedding them in the VideoPoints interface.

Sometimes, the segments may not have any keywords when all the frames in the segment contain only images. In that case, the display would be blank. The final interface after integrating the keywords module looks as shown in Figure 4.7. The user hovers on the ninth segment, and a pop up comes out containing the preview of the image and the keywords summary for the segment. The red arrow indicates the position of mouse hover.

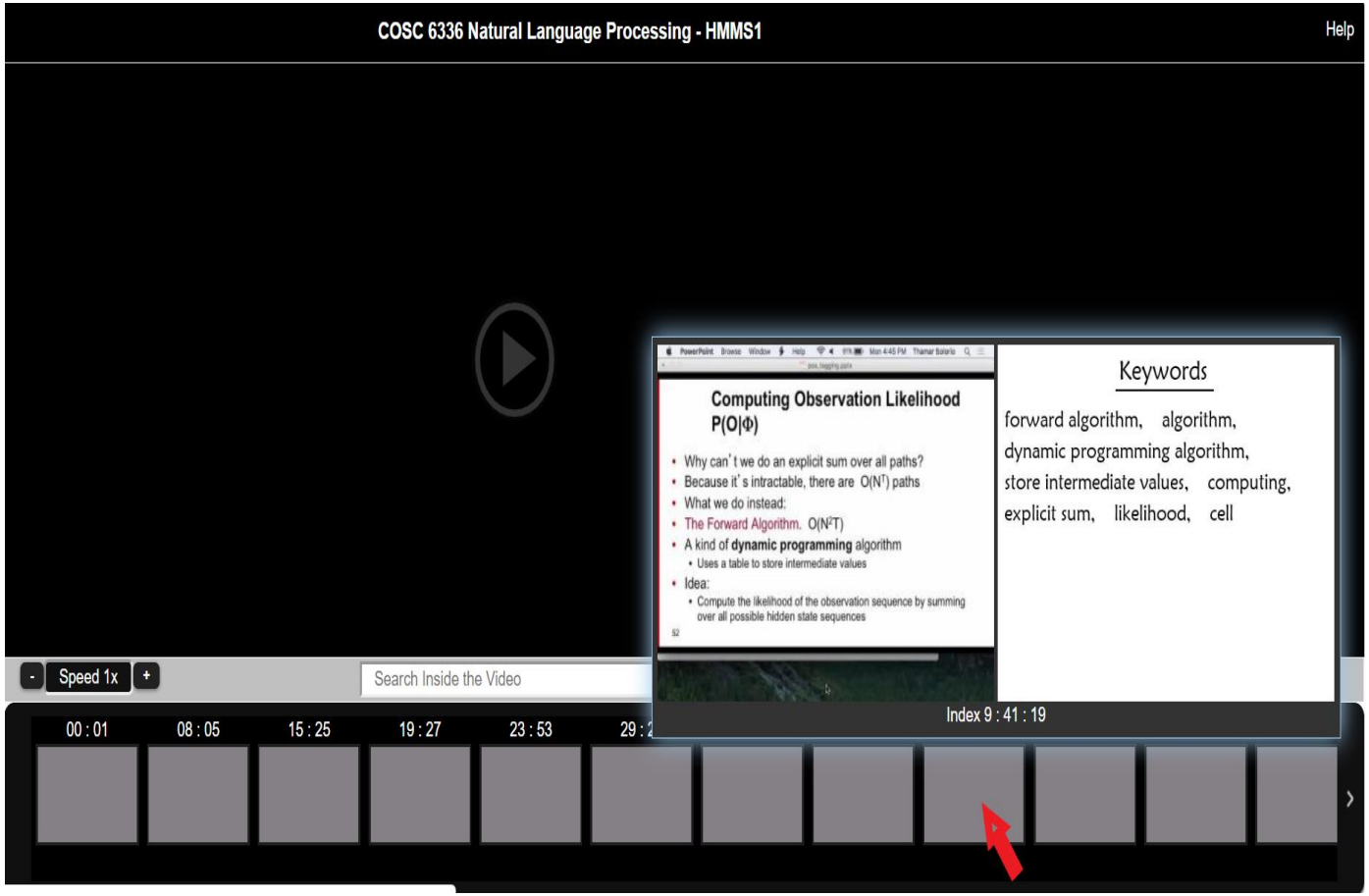

Figure 4.7 VideoPoints Interface with Keywords Summary

By looking at this, the user can determine the word ‘forward algorithm’ has higher importance followed by ‘algorithm,’ and ‘dynamic programming algorithm’.

CHAPTER – 5: ASSESSMENT

The assessment phase helps ascertain the strengths and limitations of the methodology used to extract keywords for the video lectures. This section details the tool built for the collection of ground truth and explains the process of evaluation.

5.1 Ground Truth Collection

To evaluate the performance of the keyword detection algorithm, human-picked keywords for the lecture videos were compared with the system-generated keywords. To select the appropriate keywords for each segment of the video, I took the assistance of experts like the instructors who are well-versed with the subject, or students who have been part of the course.

To accomplish this task, I have created an interface which gives experts the flexibility to determine the keywords in different segments. As mentioned before, the keywords extracted would differ when the indexing algorithm changes. Here I assumed that the indexing algorithm had produced good index points and proceeded to the next steps.

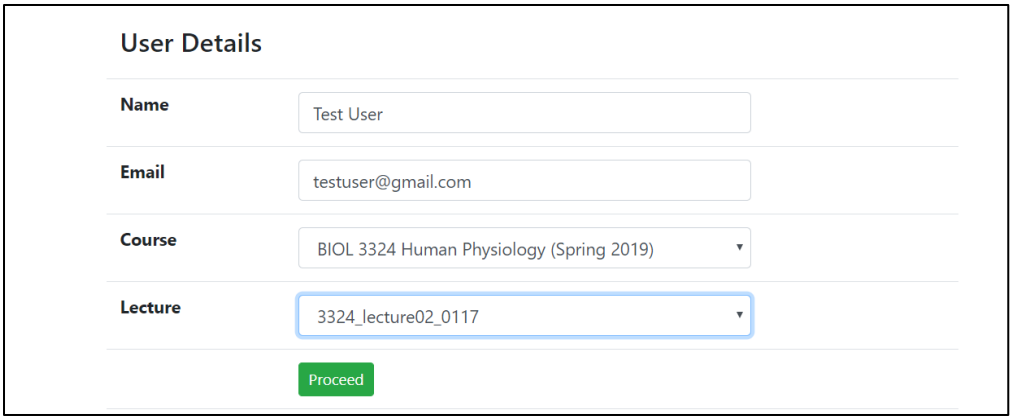

Figure 5.1 Login Screen for Ground Truth Collection

Figure 5.1 represents the login screen for ground truth collection. The users who provided the ground truth entered their Name and Email. I pre-populated the relevant courses and their corresponding lectures. The users selected the course and the lecture they were interested in tagging and clicked 'Proceed.' The users landed on a page where they provided the ground truth for a video lecture. Figure 5.2 shows the ground truth interface with its components labeled 1 – 7.

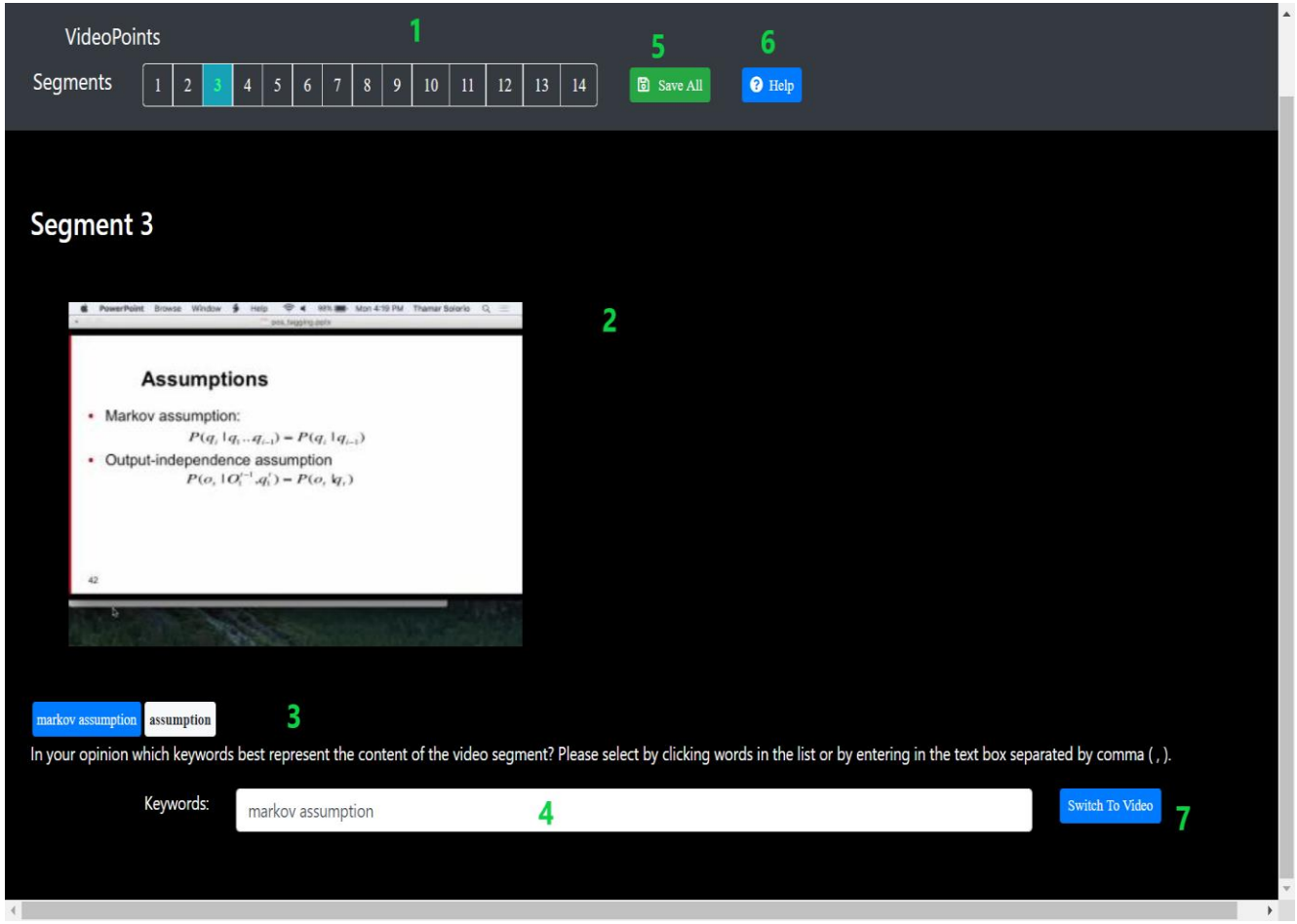

Figure 5.2 Ground Truth Interface

Ground truth interface gives access to the user to provide the feedback. This interface has been designed to reduce the burden on the user in manually tagging the video with the relevant set of keywords.

The components present on the screen marked with numbers in Figure 5.2 are as follows:

1: It represents the total number of segments present in the video lecture. The user can click on any of the segment numbers and navigate to the corresponding segment of the video. This component gives flexibility to the user to identify keywords for a part of the lecture.

2: This component includes all the slides or transition points of a segment. The user can scroll to left or right to access the relevant slides.

3: All potential keywords generated by the algorithm are shown as buttons. The user can click the corresponding button if he/she finds the keyword in the set presented. There are three different types of buttons on the screen.

Table 5.1 Color Codes of Buttons

Button Type	Significance
Button with Blue background	The button appears with a blue background when the user has selected potential keyword as a relevant keyword for the segment. The user can deselect it by clicking the button again.
Button with Bold-faced letters	It is an additional feature which highlights the buttons containing potential keywords that are present on the slides the user can currently see (viewport) without scrolling to the left/right. Buttons already with blue background do not change.
Button with Normal text	These are keywords which are not present on the slides in the current viewport. Buttons already with blue background do not change.

4: It is a textbox which shows the keywords selected by the user. If the user does not find the keyword he/she wants to pick as part of the pre-generated words, he/she can add them by editing the textbox.

5: 'Save All' button saves the work done by the user

6: The button ‘Help’ acts as an assistance tool, guiding the user about the features present on the interface. The interface also provides all the instructions during the screen launch. **7:** Clicking the button ‘Switch To Video’ plays the video relevant to the segment. With this option, the user can directly watch the video without navigating to the VideoPoints website [27].

The users can save their work and re-login at any time to continue with their work. This ensures that the user does not have the requirement to complete video tagging in one shot.

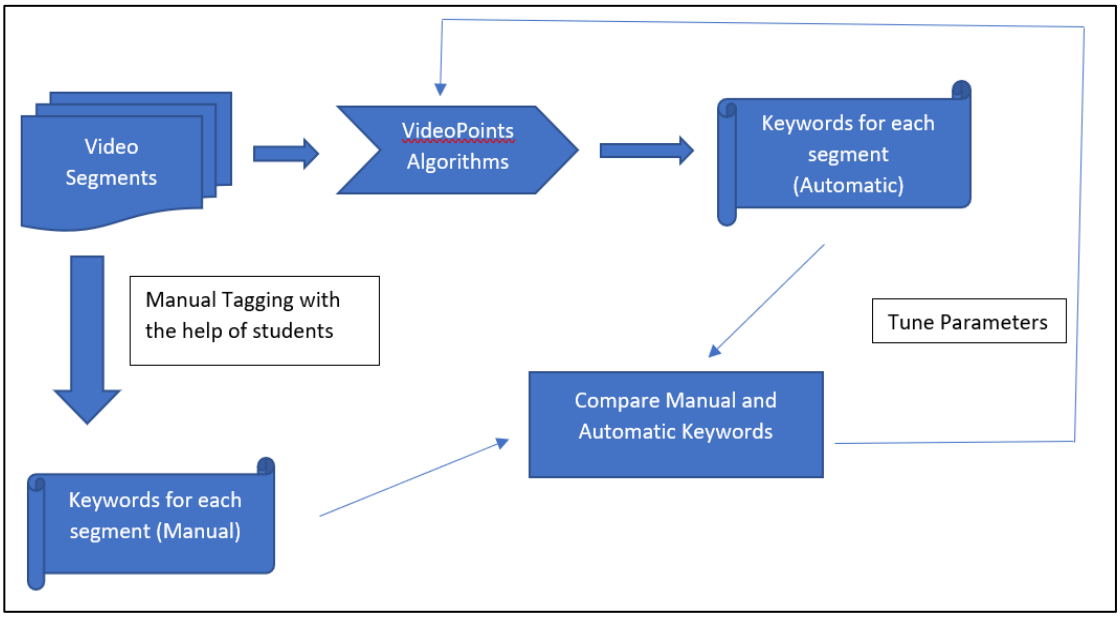

Figure 5.3 Utilizing Ground Truth to Tune Parameters

Figure 5.3 depicts the process of using the ground truth provided by experts to make the results better. The process of keyword extraction involves different parameters like *stopwordRemoval*, *applyStemming*, *reduceRareWordWeights*, *assignDomainImportance*, *applyISF*, which can take boolean values. There are other parameters like frequency weight, font weight, and time weight that can take values ranging from 0-1. The pre-

generated keywords in the ground truth interface are the keywords generated by one set of values assigned to the parameters and weights. After the users give feedback, I compared human-picked keywords with the keywords generated by different sets of values assigned to the parameters and determined the best set of values.

5.2 Metrics for Evaluation

The process of keyword extraction involves classifying a word into either of the two classes: keyword or not a keyword. Since I am interested in the class of identifying keywords and determine how many keywords presented to the user are actually useful, accuracy may not be a suitable measure. I employ well-known metrics *Precision* and *Recall* for evaluation. These measures are widely represented using a *Confusion Matrix*, which contains information about the actual and the predicted results. In my application, actual words represented the keywords given by the user and predicted words are the keywords generated by the algorithm.

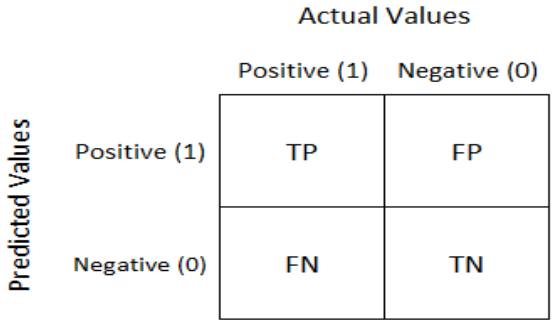

Figure 5.4 Confusion Matrix

Figure 5.4 represents the confusion matrix. The terms TP, FP, FN, and TN are defined as follows:

1. TP stands for True Positive. It is the count of words identified as keywords both by the user and the algorithm.
2. FP stands for False Positive. It is the count of words which are identified by the algorithm as keywords and not by the user.
3. FN stands for False Negative. It is the count of words which are identified by the user as keywords and not by the algorithm.
4. TN stands for True Negative. It is the count of words which are not identified both by the user and the algorithm.

Using this terminology, precision and recall are defined in the following sections.

5.2.1 Precision

Precision is the ratio of the number of correctly predicted positive examples to the total number of predicted positive examples.

$$Precision = \frac{TP}{TP + FP}$$

5.2.2 Recall

Recall is the ratio of the number of correctly predicted positive examples to the total number of actual positive examples.

$$Recall = \frac{TP}{TP + FN}$$

5.2.3 F1-Score

It is the measure of the accuracy of the keywords predicted by the algorithm. Often, a higher precision may lead to a lower recall, and a higher recall may lead to a lower precision. To balance the precision and recall scores, F1-measure is used as a standard to measure the accuracy. It is the harmonic mean of precision and recall.

$$F1 - score = \frac{2 * precision * recall}{precision + recall}$$

5.2.4 Partial Precision Score

The True Positives in the precision only accounts for the perfect matches in keywords generated by the algorithm and human-picked keywords. For n-grams, even if one word of the n-gram goes wrong, the entire word is pruned giving it a score of 0. To account for near-misses, I computed the partial match scores for each n-gram predicted by the algorithm and the user. The partial match score of an n-gram is the ratio of the number of words common in the predicted n-gram and user selected n-gram to the number of words in the union set of the n-grams.

If A and B are the n-grams generated by an algorithm and human, respectively, the partial match score is as follows:

$$Partial\ Match\ Score(PMS)\ =\ \frac{count\ of\ words(A \cap B)}{count\ of\ words(A \cup\ B)}$$

The partial precision score of a segment is equal to the average of partial match scores of words generated by the algorithm.

$$Partial\ Precision\ Score\ =\ \frac{Sum\ of\ PMS\ for\ words\ generated\ by\ algorithm}{Total\ no.\ of\ words\ predicted\ by\ algorithm}\ .$$

5.2.5 Partial Recall Score

Like the partial precision score, I computed the partial recall score. The partial recall score in a segment is equal to the average of partial match scores of words generated by the user.

$$Partial\ Recall\ Score\ =\ \frac{Sum\ of\ PMS\ for\ words\ selected\ by\ user}{Total\ no.\ of\ words\ selected\ by\ user}\ .$$

5.2.6 Partial F1 Score

This is like the F1 score, but it considers Partial Precision Score and Partial Recall Score for computation.

$$F1-score\ =\ \frac{2 * partial\ precision * partial\ recall}{partial\ precision + partial\ recall}\ .$$

5.2.7 BLEU Score

BLEU score, which stands for Bilingual Evaluation Understudy is a scoring metric to evaluate the candidate keyword with a set of reference keywords. Here the candidate keyword is the word generated by the algorithm and reference keywords are the words selected by humans. This metric also accounts for the ordering of words in the candidate and the reference n-grams. The ordering is usually not taken care in standard metrics like precision and recall. BLEU score can also be termed as modified precision. It outputs a value between 0 and 1. A value closer to 1 indicates that the candidate keyword is more similar to the reference keywords.

5.2.8 Fleiss' Kappa Score

Fleiss' Kappa score is a statistical measure for assessing the reliability of agreement among a fixed number of raters when assigning ratings to several classifying items [28]. It varies from -1 to 1. A score of 1 indicates perfect agreement and a score of -1 indicates perfect disagreement. This score tells how well the users agree with each when picking the keywords for the same segment.

5.3 Study of Parameters

I gathered keywords for 121 segments of 11 videos belonging to Computer Science, Biology and Biochemistry from 16 different students and tutors. More than one user rated most of the segments. Table 5.2 shows the count of raters tagging different segments.

Table 5.2 Count of raters for different segments

Total Segments	Total Raters
3	9
3	8
4	7
2	6
7	3
80	2
22	1

Table 5.2 shows that more than five users tagged 12 segments, three users rated seven segments, two users tagged 80 segments, and one user tagged the rest of the segments.

Using the keywords given by raters for various segments, I computed the scores of different metrics using a random combination of weights and values as parameters to my algorithm.

The total number of keywords presented as a summary of each segment varied between 7 and 20. It means each segment displayed at least 7 words and at most 20 words. I decided

the minimum and the maximum number of words to display on each segment by observing a few lectures. The range set by the keyword detection algorithm defines the final count on how many words should be displayed. I have considered the words which have scores above 60% of the range value, 50% of the range value, and 40% of the range value in each segment and computed the results.

I altered the weights of the parameters like frequency, font and time and assigned different combinations of binary values to *stopwordRemoval*, *applyStemming*, *applyISF*, *assignDomainImportance*, and *reduceRareWordWeights*.

Table 5.3 Variation of Parameters to Generate Different Sets of Keywords

Parameter	Set-1	Set-2	Set-3	Set-4	Set-5
Frequency Weight	0.5	0.5	0.5	0.4	0.4
Font Weight	0.2	0.4	0.4	0.5	0.3
Time Weight	0.3	0.1	0.1	0.1	0.3
Stop Word Removal	True	True	True	True	True
Apply Stemming	True	True	True	True	True
Assign Domain Importance	True	True	True	True	True
Reduce Rare Word Weights	True	True	False	True	False
Apply ISF	True	True	True	True	True

I have experimented with Table 5.3 which represents the top five performing set of values. From the above table, I inferred the values for any set. For instance, Set-1 keywords used

the frequency weight as 0.5, font weight as 0.2, time weight as 0.3, and the parameters *stopwordRemoval*, *reduceRareWordWeights*, *applyStemming*, *assignDomainImportance*, and *applyISF*, as TRUE.

Using different sets of values, I generated keywords and computed the metrics discussed in Section 5.2. As ground truth has more than one user rating the segments, I calculated the scores by two methods.

5.3.1 Keywords from All Users

In this method, the gold set of keywords for a video segment contained the keywords given by all users tagging the segment.

Table 5.4 Sample Keywords given by Users for a Segment

Rater	Keywords
User 1	{Word 1, Word 2, Word 3}
User 2	{Word 1, Word 3, Word 4}
User 3	{Word 1, Word 2, Word 3, Word 5, Word 6}

Table 5.4 shows sample keywords tagged by different users for one segment of the video. Now the gold set for this segment is {Word 1, Word 2, Word 3, Word 4, Word 5, Word 6}. I compared the gold set with the keywords generated by the algorithm.

As mentioned before, I restricted the number of keywords generated by algorithm between 7 and 20 and experimented by considering keywords that have the score above 60% of the

range, 50% of the range and 40% of the range. Figure 5.5 – Figure 5.10 shows results considering different ranges of keywords.

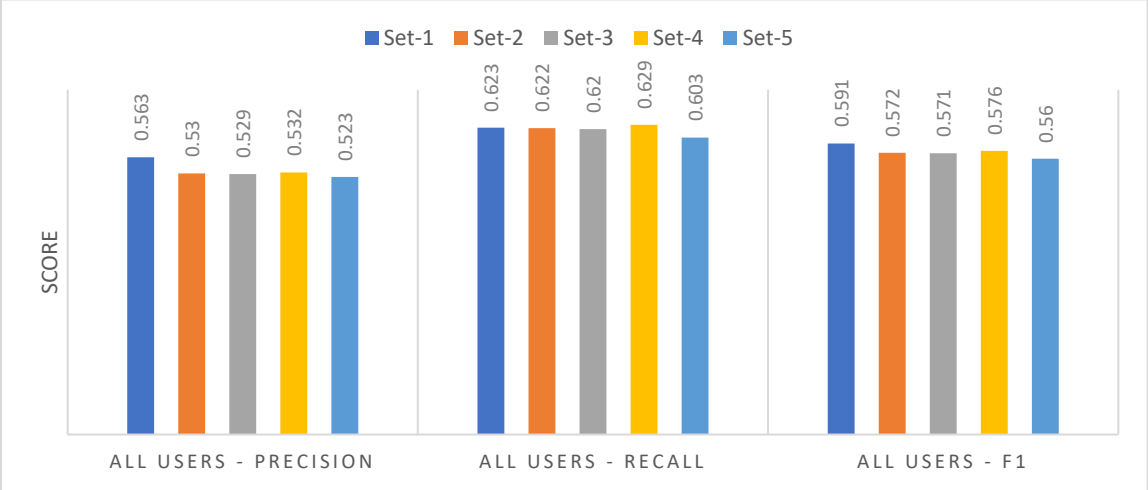

Figure 5.5 Scores with Keywords above 60% of the Range – All Users

From Figure 5.5, Set-1 gives the highest F-1 score which is equal to 59.1%. The values of precision and recall achieved by Set -1 are 56.3% and 62.3%, respectively.

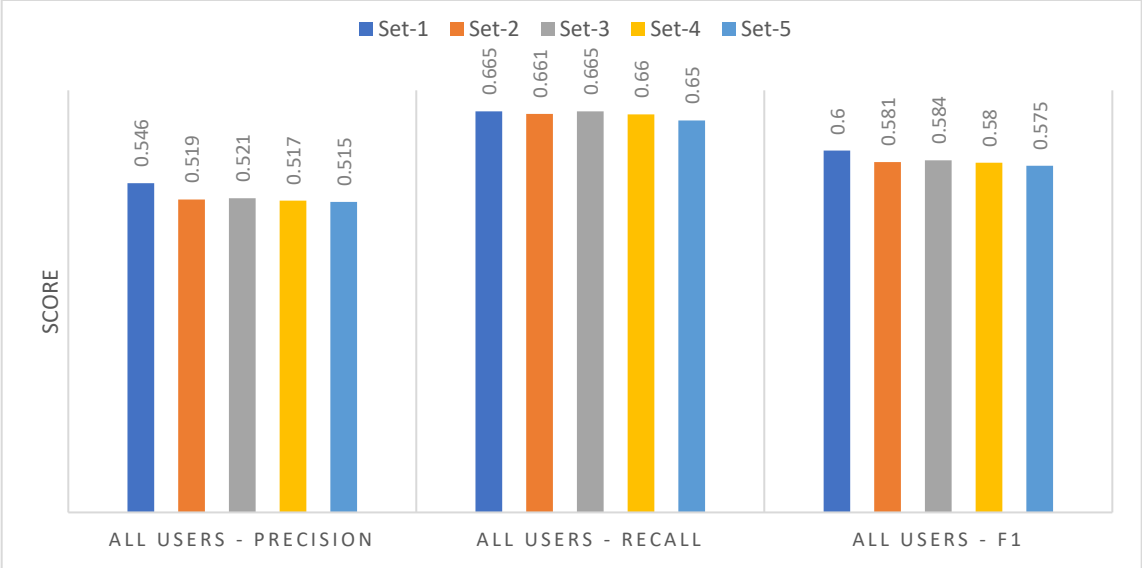

Figure 5.6 Scores with Keywords above 50% of the Range – All Users

From Figure 5.6, Set-1 gives the highest F-1 score which is equal to 60%. The values of precision and recall achieved by Set -1 are 54.6% and 66.5%, respectively.

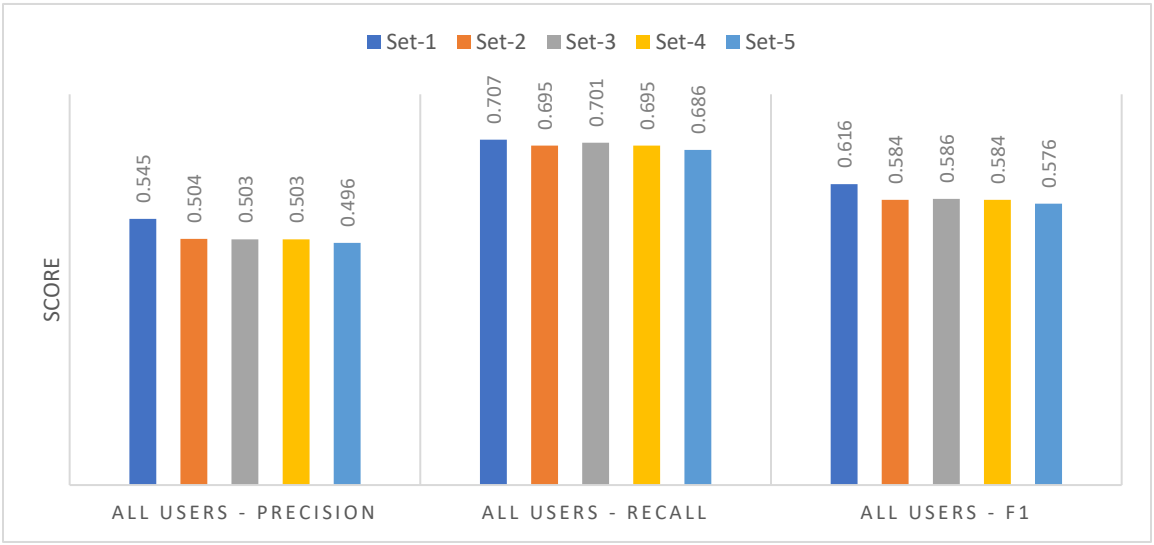

Figure 5.7 Scores with Keywords above 40% of the Range – All Users

From Figure 5.7, Set-1 gives the highest F-1 score which is equal to 61.6%. The values of precision and recall achieved by Set -1 are 54.5% and 70.7%, respectively.

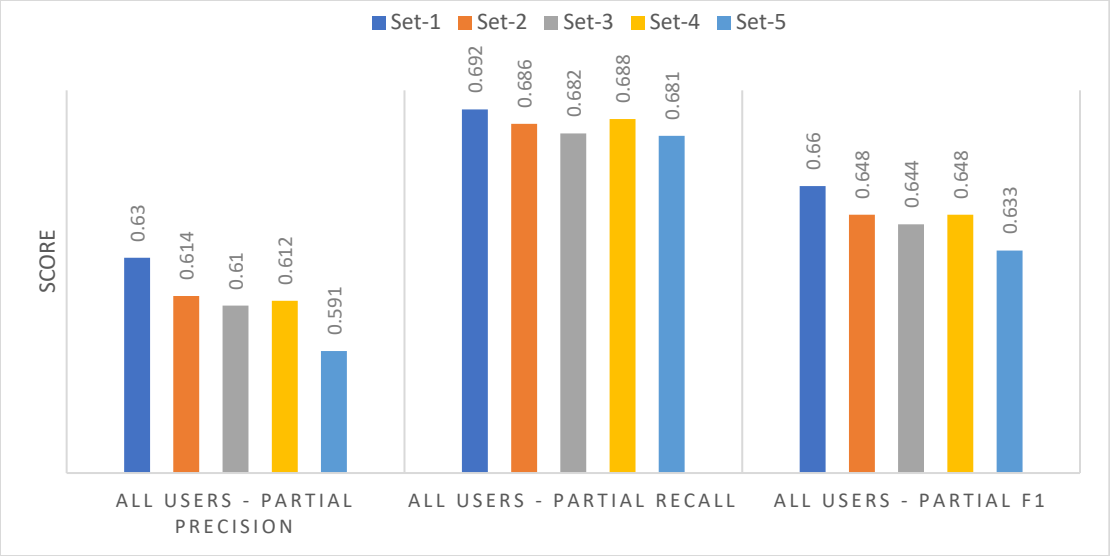

Figure 5.8 Partial Scores with Keywords above 60% of the Range – All Users

From Figure 5.8, Set-1 gives the highest F-1 score which is equal to 66%. The values of precision and recall achieved by Set -1 are 63% and 69.2%, respectively.

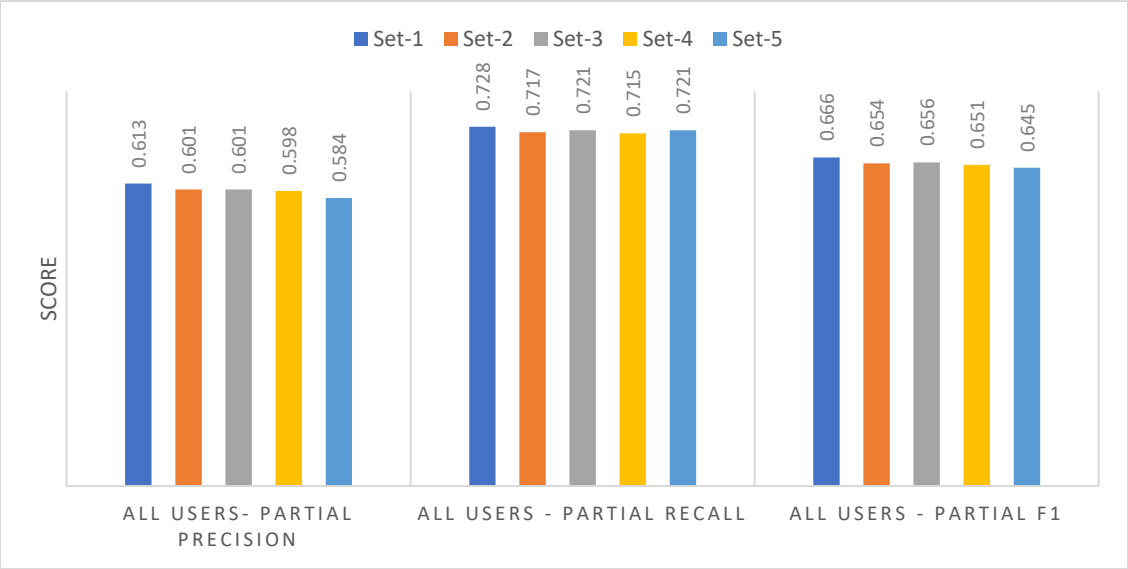

Figure 5.9 Partial Scores with Keywords above 50% of the Range – All Users

From Figure 5.9, Set-1 gives the highest F-1 score which is equal to 66.6%. The values of precision and recall achieved by Set -1 are 61.3% and 72.8%, respectively.

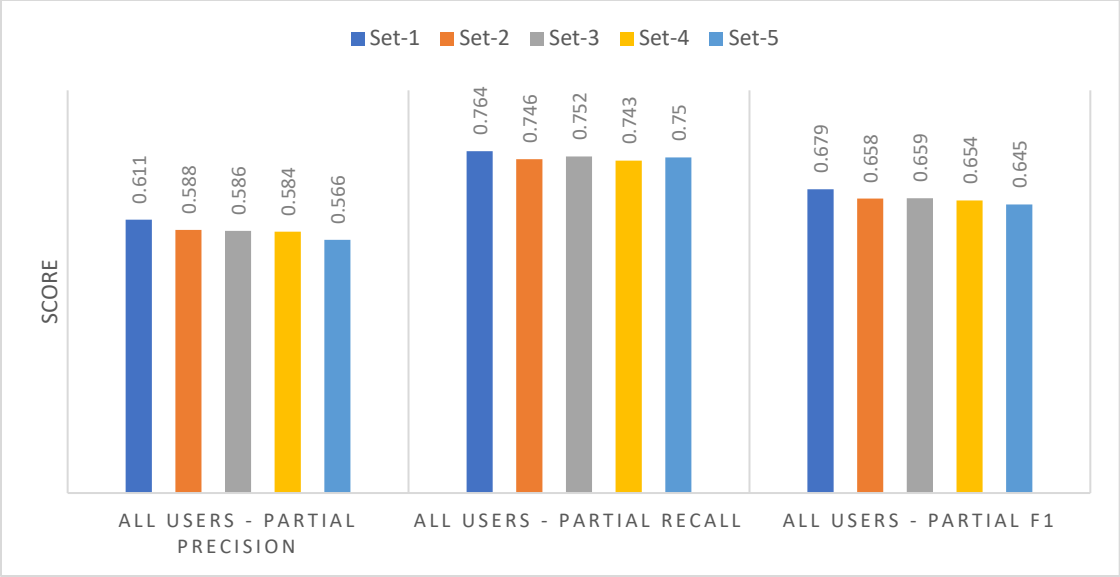

Figure 5.10 Partial Scores with Keywords above 40% of the Range – All Users

From Figure 5.10, Set-1 gives the highest F-1 score which is equal to 67.9%. The values of precision and recall achieved by Set -1 are 61.1% and 76.4%, respectively.

5.3.2 Keywords from Majority Users

In this method, the gold set of keywords for a video segment contained the keywords that were selected by at least half of the users tagging the segment. (see the beginning of the previous Section 5.3.1)

From Table 5.4, since there are three users, I considered keywords which are commonly picked by more than or equal to ½ * 3 ≈ 2 users. The final list of keywords comprised of three words {Word 1, Word 2, Word 3}, as they are rated by two or more users.

Considering this, I compared the gold set against the algorithm generated words and obtained the results as shown below:

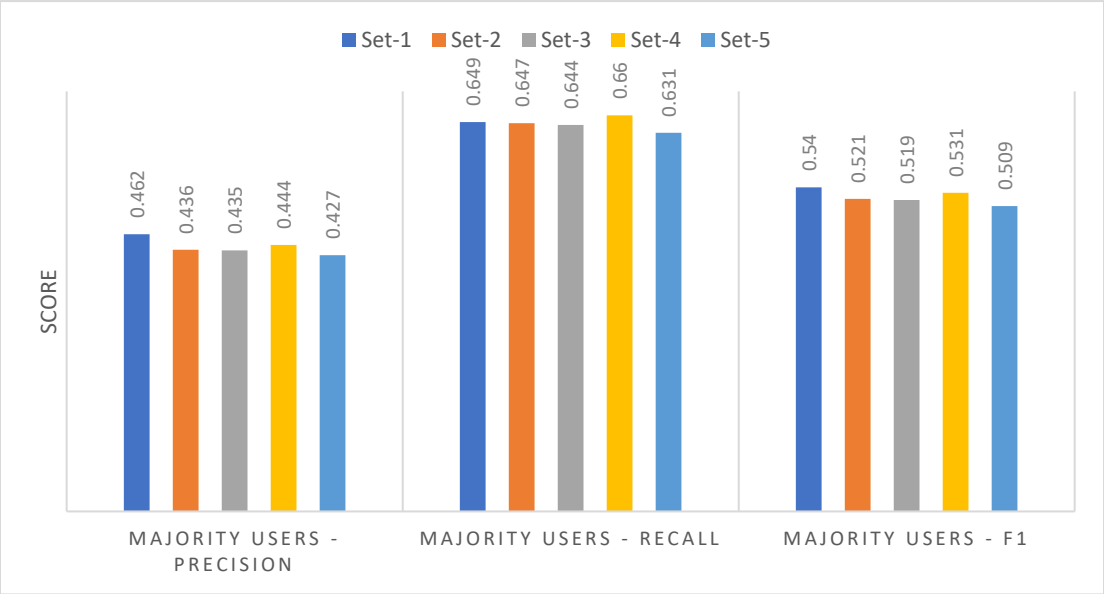

Figure 5.11 Scores with Keywords above 60% of the Range – Majority Users

From Figure 5.11, Set-1 gives the highest F-1 score which is equal to 54%. The values of precision and recall achieved by Set -1 are 46.2% and 64.9%, respectively.

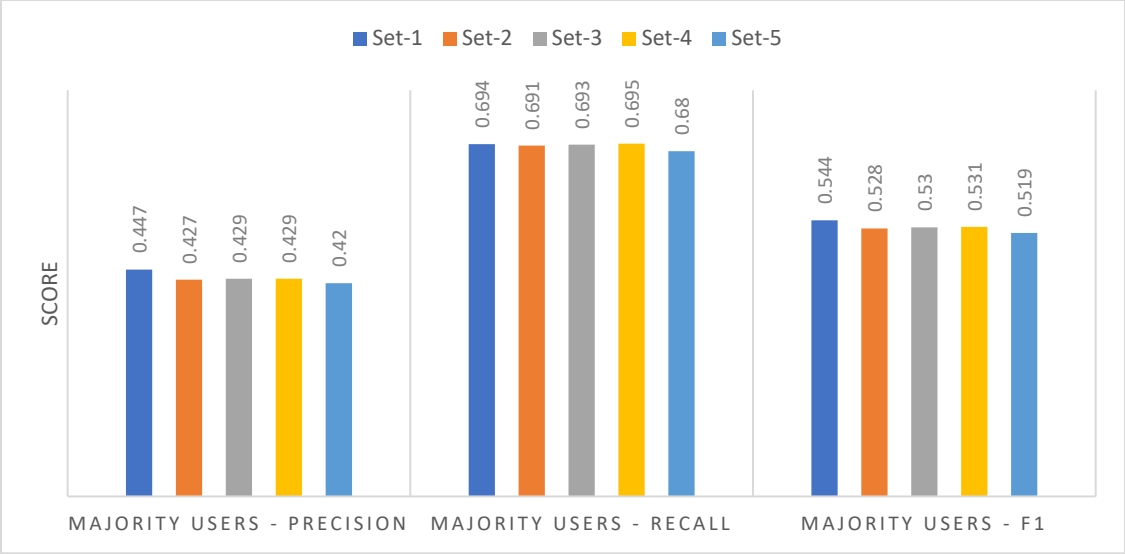

Figure 5.12 Scores with Keywords above 50% of the Range – Majority Users

From Figure 5.12, Set-1 gives the highest F-1 score which is equal to 54.4%. The values of precision and recall achieved by Set -1 are 44.7% and 69.4%, respectively.

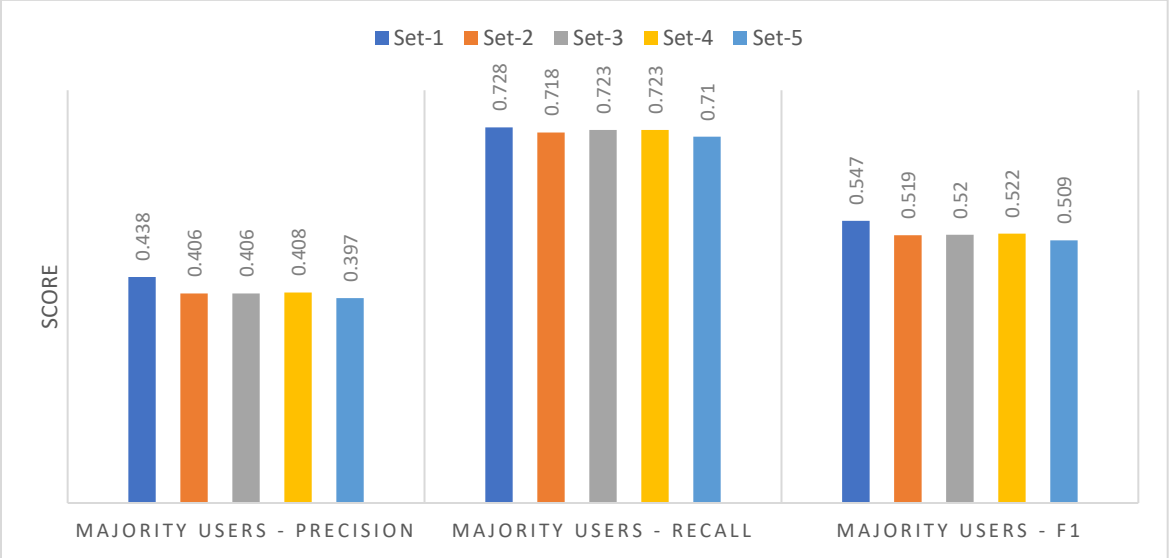

Figure 5.13 Scores with Keywords above 40% of the Range – Majority Users

From Figure 5.13, Set-1 gives the highest F-1 score which is equal to 54.7%. The values of precision and recall achieved by Set -1 are 43.8% and 72.8%, respectively.

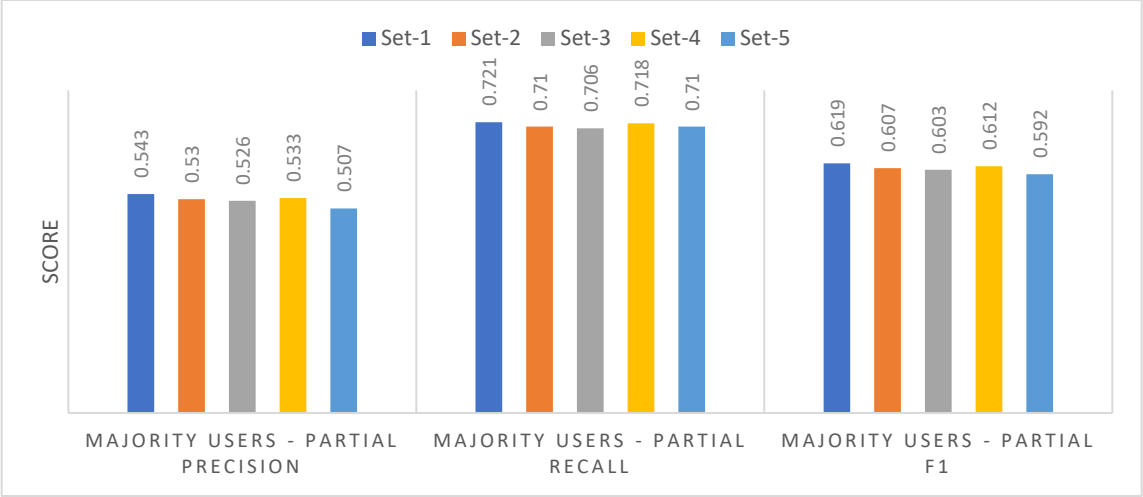

Figure 5.14 Partial Scores with Keywords above 60% of the Range – Majority Users

From Figure 5.14, Set-1 gives the highest F-1 score which is equal to 61.9%. The values of precision and recall achieved by Set -1 are 54.3% and 72.1%, respectively.

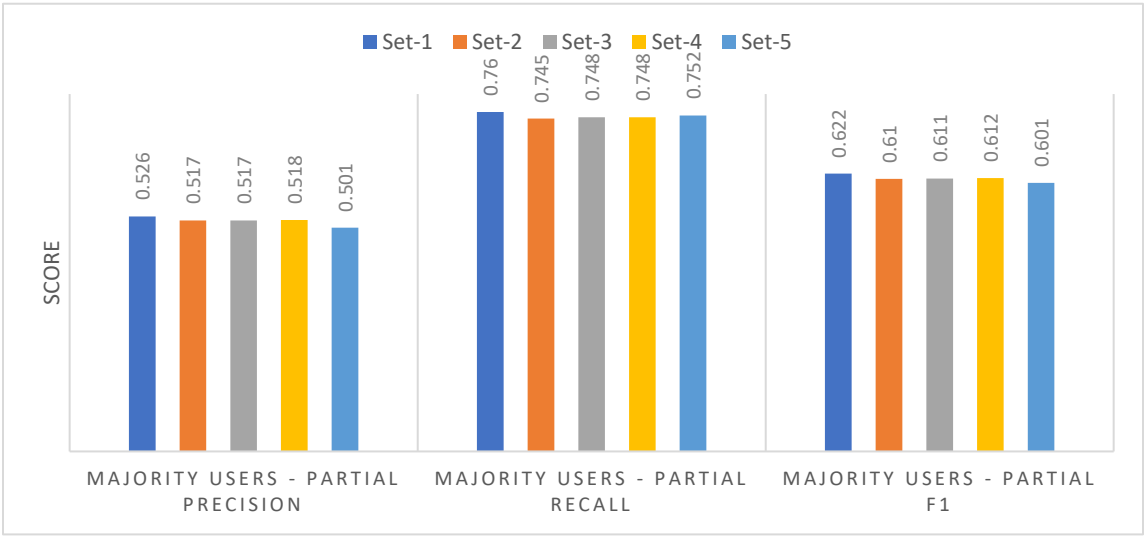

Figure 5.15 Partial Scores with Keywords above 50% of the Range – Majority Users

From Figure 5.15, Set-1 gives the highest F-1 score which is equal to 62.2%. The values of precision and recall achieved by Set -1 are 52.6% and 76%, respectively.

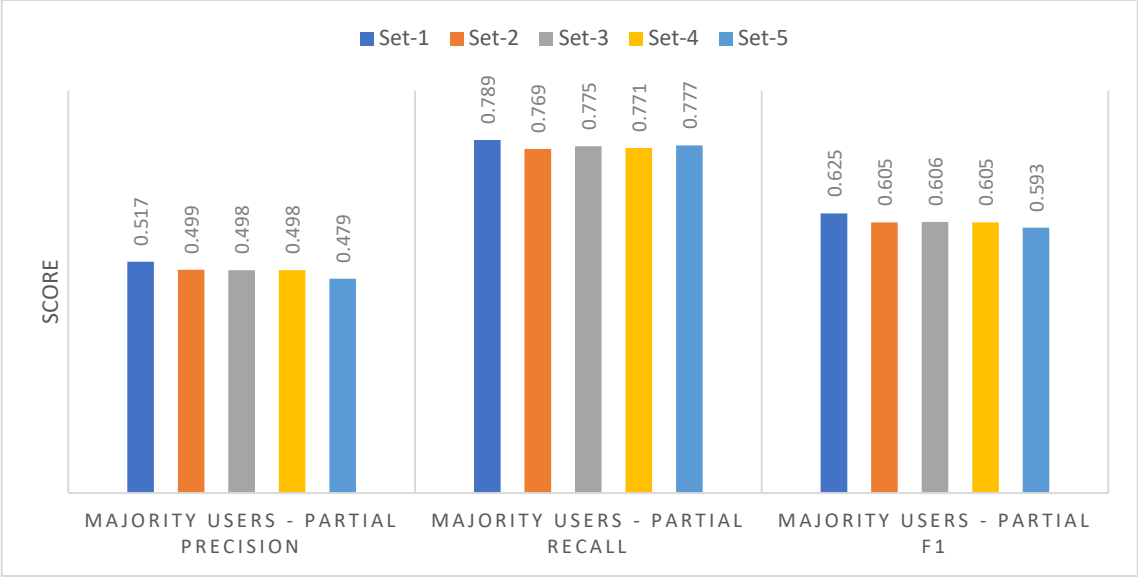

Figure 5.16 Partial Scores with Keywords above 40% of the Range – Majority Users

From Figure 5.16, Set-1 gives the highest F-1 score which is equal to 62.5%. The values of precision and recall achieved by Set -1 are 51.7% and 78.9%, respectively.

By observing results obtained by different combinations of parameters for 121 segments, I determined that Set-1 performs relatively well. Table 5.5 shows the values of final values assigned to the parameters of the keyword detection algorithm.

Table 5.5 Final Parameters of the Keyword Detection Algorithm

Parameter	Value
Frequency Weight	0.5
Font Weight	0.3
Time Weight	0.2
Apply Stop Word Removal	True
Apply Stemming	True
Assign Domain Importance	True
Reduce Wts of Rare Words	True
Apply ISF	True

5.4 Summary of Results

Assigning the values of Set–1 to the keyword detection algorithm generated the best results. Figure 5.17 shows different metrics obtained by Set-1 values with varying range. By analyzing the graph, the most common pattern observed was the number of keywords selected by the algorithm increased, values of precision and partial precision decreased. Recall and partial recall increased as the keywords selected by the algorithm increased. Overall, F1 and partial F1 scores increased with an increasing number of keywords. The highest value of F1 obtained was 61.5% in the strict evaluation and 67.8% in the partial evaluation by considering keywords given by all users whose scores were 40% above the range.

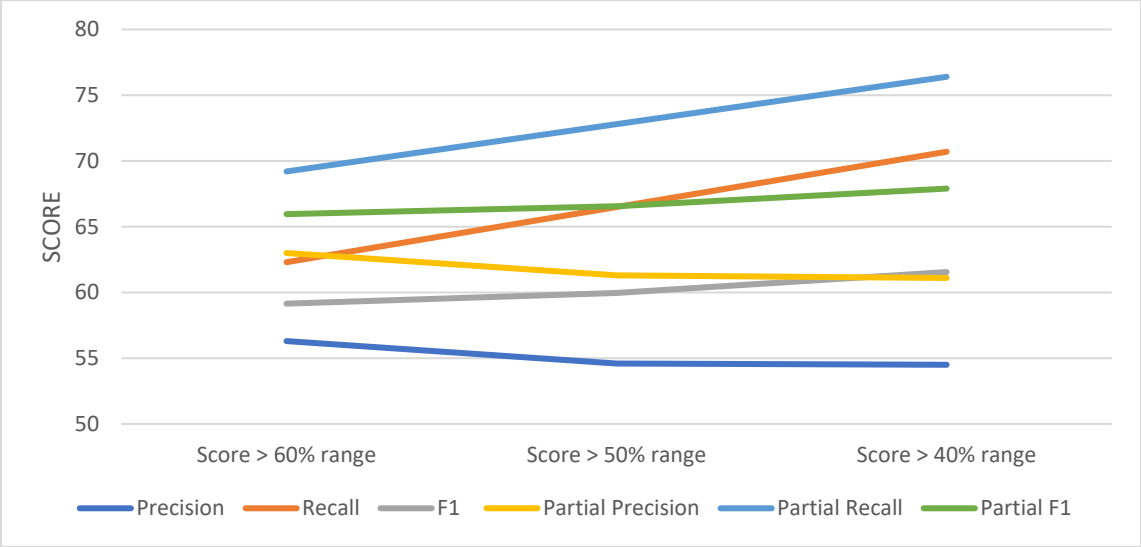

Figure 5.17 Scores from Set-1 Parameters with a Varying Range

5.4.1 BLEU Score Analysis

I also calculated the BLEU score using Set-1 parameters. It slightly decreased when compared with the highest values attained with another set of parameters. The BLEU score decreased with an increasing number of keywords. Figure 5.18 can illustrate this pattern. BLEU score also followed the pattern followed by the precision metric, however, the values of BLEU scores were higher compared to the precision and slightly lower compared to the partial precision. This is a good sign that the n-grams predicted by the keyword detection algorithm follow the order as expected by users.

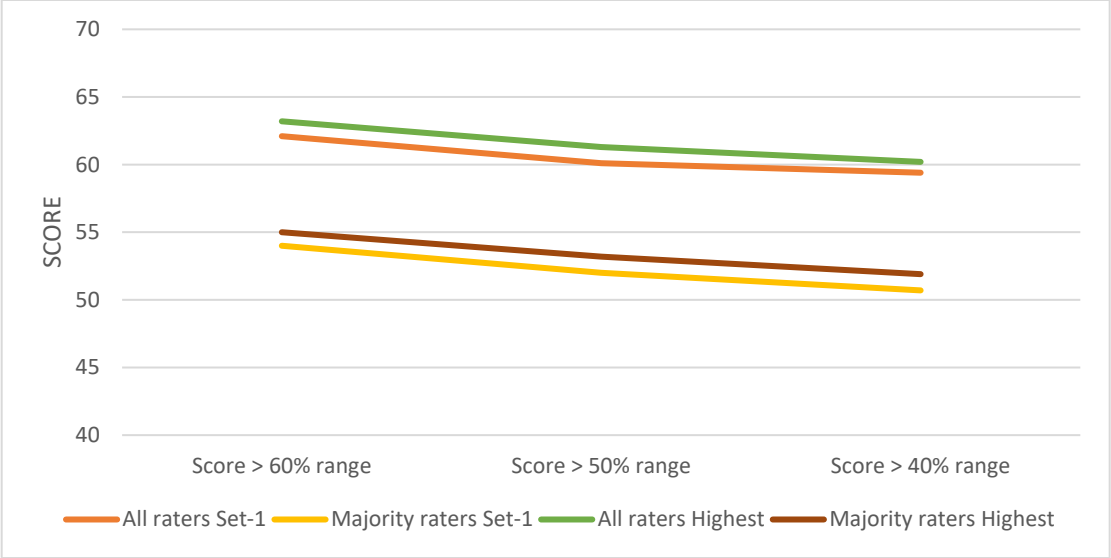

Figure 5.18 Comparison of BLEU Score: Set-1 Values and Highest Values

5.4.2 Rater Agreement Analysis

To measure the agreement among different raters, I computed the Fleiss' Kappa score for segments having more than three raters. The scores obtained were in the range of 0 – 0.2, which indicate there was only a slight agreement among the raters. Table 5.6 shows the results obtained for the rater agreement analysis.

Table 5.6 Rater Agreement Score

Total Keywords	Raters	Kappa Score
97	9	0.133
108	8	0.111
82	7	0.172
84	6	0.153

These low scores indicate that the users did not agree with each other. During the ground truth collection process, I did not put a limit on the number of keywords selected by the user for each segment. If one user selected three keywords for a segment, and another user selected ten keywords for the same segment, then the maximum number of keywords that can be common for both the users was three, and the metric considered this as a high disagreement. This is the main reason for obtaining low rater agreement scores.

5.5 Analysis of Errors

In this section, I analyzed the output of keyword detection algorithm and identified the vulnerabilities of different techniques that could be potential reasons for generating incorrect keywords.

5.5.1 OCR Issues

As mentioned before, the process of keyword extraction starts with erroneous text. As OCR technology plays a key role in determining the text on video frames, the process should not be erroneous. Several big giants like Microsoft and Google are working on OCR to make its output error-free. They have been training the OCR on billions of images to get the best of it. So, with a good OCR in place, the system can avoid most of the false detection of words.

Another issue with OCR is that it does not detect punctuations and sentence boundaries. It just scans the text left to right, top to bottom. This leads to a collection of words from the frames rather than having sentences. The contextual information is also not present, which is very critical for the application of Natural Language Processing techniques. Without the contextual information, the system treats the words as a bag-of-words model for generating frequencies. This may not give the best outcome. Also, if a few words from different sentences form an n-gram, and if the n-gram is valid, this increases the room for errors.

I applied some techniques to correct OCR errors. But since I do not have the context exactly captured, there is no guarantee for error correction. The second phase of correction identifies valid and invalid words based on word frequencies. If OCR identifies word 'w_1' as 'w_2' at all the occurrences of 'w_1,' then it cannot be corrected due to its high frequency. This mainly happens with real-word errors.

5.5.2 Issues with External sources

In the process of identifying valid n-grams, I am making use of the PhraseFinder API [18], a search engine for Google n-grams dataset. There is a rough estimate of the volumes of books Google scanned to muster the database, but there is no information provided on the category of books and their domains. It is possible that it has not scanned enough books of a particular domain. The n-gram database has a restriction which says words should occur in at least 40 books to get an entry into the database [19]. When resources belonging to a domain are sparse, some domain-specific words may lose the chance of getting into the database, and this makes the global count 0. The system may lose some important information due to this. Other external sources include reference dictionaries from Oxford for identifying domain-specific words and stop word list. The stop word list almost contains most of the frequently used stop words, and there is a low chance to get errors with this source. But the dictionaries from Oxford are updated frequently with new words. To reduce the chance of errors, I should update them frequently.

5.5.3 Limitations of Cleaning Extraneous Text

In the process of cleaning extraneous text, I set a threshold dynamically to identify the words repeating at similar positions and discarded them. There may be cases, where extraneous text might appear in very few transition points, and their total occurrences might not satisfy the threshold criteria. This makes them a part of the possible keywords.

Another possibility of error occurs when the instructor uses the same title name on most of the slides. As this title constantly appears in a similar position, the algorithm may discard some essential words. In this case, the current system does not select them as keywords as it discards them.

5.5.4 Keywords from Video Transcript

Instructors may emphasize a specific application of the topic during the lecture, and this may not be a part of the presentation. Since I only considered the text from the frames rather than the speech transcript, the current methods tend to miss those words as their weight might be relatively low. The expert who is picking the tags may watch the video and select the words that are exclusive to the audio. This is another possible case of missing keywords.

CHAPTER – 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

Availability of keywords for video segments is likely to have a significant impact on students' learning outcome as they improve navigation across a lecture. Processing the keywords manually is a complicated task. Determining the word importance based on the frequency of word may not result in an efficient keyword summary. This thesis work proposed and developed methods to automatically generate keywords by assigning different weights to various properties of text like the font characteristics of the word, the time presence of the word, besides the frequency of the word. The frequency internally considered many other steps like removing stop words, applying stemming, assigning domain importance, pruning the weights of rare words, and applying TF-ISF. Post-processing was done on the OCR output to rectify errors using spell correction techniques before extracting different features from the words. An interface was also developed to gather the necessary ground truth from the students to evaluate the results obtained by the proposed methods. I evaluated the keywords generated by different versions of the algorithm with the feedback given by the users, and my methods achieved a precision of 54.5%, recall of 70.7% and F1 score of 61.6% in the strict version. These scores were also computed by relaxing the metrics to account for partial matches and achieved a precision of 61.1%, recall of 76.4% and F1 score of 67.9%. I have analyzed the number of words to display as part of the summary for each segment and decided that considering the words

which have a score higher than 40% of the range would give the best results. I also restricted the maximum allowable words in a segment to 20.

6.2 Future Work

The data from the video frames may not adequately present the information on the topic. A hybrid speech and text model can be developed to determine the keywords. In my experience, speech preserved the semantics well compared to the OCR. Future improvements in ASR technology will automatically lead to improved keyword detection. The *assignDomainImportance* parameter identified words belonging to a specific domain and boosted their weights. However, the lecture might belong to two or more domains and improvement can be made to the current methods to handle this. I multiplied the weights by a factor 2^k, which worked well. In the future, other researchers could revisit this area and determine a more reasonable factor. More ground truth can be gathered to have the same segment tagged by multiple users and rank the words based on the user agreement to compare them against the ranking generated by the current methods and measure deviations. This paves a way in working towards achieving minimum deviation. Considering these methods might help achieve higher F1 score improving the precision and recall and generate high-quality keywords.

REFERENCES

[1] V. Koller, S. Harvey and M. Magnotta, "Technology-based learning strategies," [Online]. Available: https://www.doleta.gov/reports/papers/tbl_paper_final.pdf. [Accessed 20 12 2018].

[2] J. Subhlok, O. Johnson, V. Subramaniam, R. Vilalta and C. Yun, "Tablet PC video based hybrid coursework in computer science: report from a pilot project," *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education,* vol. 39, pp. 74–78, 2007.

[3] T. Tuna, J. Subhlok and S. Shah, "Indexing and Keyword Search to Ease Navigation in Lecture Videos," *Proceedings of the 2011 IEEE Applied Pattern Workshop,* pp. 1-8, 2011.

[4] V. Varghese, "Development and Evaluation of Textbased Indexing for Lecture Videos," M.S. Thesis, University of Houston, Houston, 2014.

[5] T. Tuna, J. Subhlok, L. and Barker, V. Varghese, O. Johnson and S. Shah, "Development and Evaluation of Indexed Captioned Searchable Videos for STEM Coursework," *Proceedings of the 43rd ACM technical symposium on Computer Science Education,* pp. 129-134, 2012.

[6] A. Mishra and S. Vishwakarma, "Analysis of TF-IDF Model and its Variant for Document Retrieval," *International Conference on Computational Intelligence and Communication Networks (CICN),* pp. 772-776, 2015.

[7] S. Lee and H. Kim, "News Keyword Extraction for Topic Tracking," *Fourth International Conference on Networked Computing and Advanced Information Management,* vol. 2, pp. 554-559, 2008.

[8] R. Chakraborty, "Domain Keyword Extraction Technique: A New Weighting Method Based on Frequency Analysis," *Computer Science & Information Technology,* vol. 3, pp. 109-118, 2013.

[9] T. Tuna, J. Subhlok, L. Barker, S. Shah, O. Johnson and C.Hovey, "Indexed Captioned Searchable Videos: A Learning Companion for STEM Coursework," *Journal of Science Education and Technology,* vol. 26, no. 1, pp. 82-99, 2017.

[10] S. V. Delden, D. Bracewell and F. Gomez, "Supervised and Unsupervised Automatic Spelling Correction Algorithms," *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration,* pp. 530-535, 2004.

[11] A. Bhardwaj, F. Farooq and H. Cao, "Topic Based Language Models for OCR Correction," *Proceedings of the second workshop on Analytics for noisy unstructured text data,* pp. 107-112, 2008.

[12] B. Youssef and A. Mohammad, "OCR Post-processing Error Correction Algorithm Usign Google's Online Spelling Suggestion," *Journal of Emerging Trends in Computing and Information Sciences,* vol. 3, no. 1, pp. 90-99, 2012.

[13] A. Huang, "Similarity Measures for Text Document Clustering," *Proceedings of the Sixth New Zealand Computer Science Research Student Conference,* pp. 49-56, 2008.

[14] K. Kukich, "Techniques for Automatically Correcting Words," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377-439, 1992.

[15] G. Cormode and S. Muthukrishnan, "The String Edit Distance Matching Problem with Moves," *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 667-676, 2002.

[16] "N-gram Models," [Online]. Available: https://sookocheff.com/post/nlp/n-gram-modeling/. [Accessed 15 9 2018].

[17] "Language Model," [Online]. Available: https://en.wikipedia.org/wiki/Language_model. [Accessed 4 10 2018].

[18] "PhraseFinder," [Online]. Available: https://phrasefinder.io/documentation. [Accessed 23 03 2019].

[19] "Google Ngram Viewer," [Online]. Available: https://en.wikipedia.org/wiki/Google_Ngram_Viewer. [Accessed 7 05 2018].

[20] "Stop words," [Online]. Available: https://www.link-assistant.com/seo-stop-words.html. [Accessed 20 05 2018].

[21] A. Jivani, "A Comparative Study of Stemming Algorithms," *International Journal of Computer Technology and Applications.*, vol. 2, no. 6, pp. 1930-1938, 2011.

[22] P. M.F, "An algorithm for suffix stripping," *Readings in information retrieval*, pp. 313-316, 1980.

[23] C. Paice, "Another stemmer," *ACM SIGIR Forum,* vol. 24, no. 3, pp. 56-61, 1990.

[24] P. Martin, "Snowball: A language for stemming algorithms," 2001. [Online]. Available: http://snowball.tartarus.org/texts/introduction.html. [Accessed 25 12 2018].

[25] "Oxford Dictionaries," [Online]. Available: http://www.oxfordreference.com. [Accessed 29 07 2018].

[26] S. K. Patro and K. K. Sahu, "Normalization: A Preprocessing Stage," *International Advanced Research Journal in Science, Engineering and Technology,* vol. 2, no. 3, pp. 20-22, 2015.

[27] "VideoPoints," [Online]. Available: http://videopoints.uh.edu/public/. [Accessed 10 04 2019].

[28] "Fleiss Kappa Score," [Online]. Available: https://en.wikipedia.org/wiki/Fleiss%27_kappa. [Accessed 31 03 2019].

[29] X. Tong and D. A. Evans, "A Statistical Approach to Automatic OCR Error Correction in Context," *Proceedings of the Fourth Workshop on Very Large Corpora,* pp. 88-100, 1996.