Text Representations of Math Tutorial Videos for Clustering, Retrieval, and Learning Gain Prediction

by

Pichayut Liamthong

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

Jan 2021

APPROVED:

Professor Jacob R. Whitehill, Master Thesis Advisor

Professor Jaseph Beck, Thesis Reader

Professor Craig E. Wills, Head of Department

Abstract

With the goal of making vast collections of open educational resources (YouTube, Khan Academy, etc.) more useful to learners, we explored how automatically extractable text representations of math tutorial videos can help to categorize the videos, search through them for specific content, and predict the individual learning gains of students who watch them. In particular, (1) we devised novel text representations, based on the output of an automatic speech recognition system, that consider the frequency of different tokens (symbols, equations, etc.) as well as their proximity from each other in the transcript. Unsupervised learning experiments, conducted on 208 videos that explain 18 math problems about logarithms show that the clustering accuracy of our proposed methods reaches 85%, surpassing that of standard TF-IDF features (78% using log normalization). (2) In a video search setting, the proposed text features can significantly reduce the number of videos (up to 88% reduction on our dataset) and amount of video time (up to 82%) that users need to spend looking for desired content in large video collections. Finally, (3) in an experiment on Mechanical Turk with $n = 541$ participants who watched a randomly assigned tutorial video between a pretest & posttest, the text features and their multiplicative interactions with students' prior knowledge provide a statistically significant benefit to predicting individual learning gains.

Contents

1 Introduction **1**

2 Background **4**

2.1 Text Representations 4

2.2 Video Content Categorization & Clustering 5

2.3 Video Retrieval 5

2.4 Estimating the Effectiveness of OERs 5

3 Text Representations **7**

3.1 Speech-to-Text Transcription 8

3.2 Token Types 8

3.2.1 Individual Token 8

3.2.2 Expression Token 9

3.3 Token Count Vector 10

3.3.1 1D (No Order Dependencies) 10

3.3.2 2D (First-Order Dependencies) 10

3.4 Token Summarization Methods 11

3.5 Fixed-Size Representation (Dataset Independent) 12

4 Dataset **13**

5 Clustering: Video Categorization **15**

5.1 Feature Modifications 16

5.1.1 Binarizing Token Counts 16

5.1.2 Weighted Frequencies 16

5.1.3 Restricting the Alphabet 17

5.1.4 1D vs. 2D 17

5.2 Comparison to TF/IDF 18

6 Searching **19**

6.1 Architecture 19

6.2 Number of Videos Watched 20

6.3 Amount of Time Watched 22

7 Learning gain prediction **24**

7.1 Simple Linear Models 25

7.2 Deeper Models 26

7.2.1 Model Architectures 27

7.2.2 Implementation 27

7.2.3 Experiments and Results 28

8 Conclusion and Future Work **37**

List of Figures

1.1 Two example videos in our study. For the bottom video, Google's Speech-to-Text API extracts the text "and we're going to solve for x ok our problem is log base 3 of x minus 1 equals 4 we're good then log base B of X is equal to Y and this is equivalent to B to the Y equals X", which shows high agreement with the visual content in the video. . . 2

6.1 The decrease in time needed to find specific math content in a set of math tutorial videos. Each line shows a different text representation over different segment lengths used for evaluation. 23

List of Tables

5.1 The Clustering Percent Accuracy (%) 17

5.2 The Clustering Percent Accuracy (%) comparing 1D v.s. 2D Representations 18

6.1 The Percent Decrease in Number of Videos Watched 20

7.1 Existing Video Correlations 29

7.2 Existing Video ANOVA 29

7.3 New Video Correlations 32

7.4 New Video ANOVA 32

7.5 The Average Learning Gain of Students 35

Chapter 1

Introduction

Consider a large repository (Khan Academy, edX, etc.) of open educational resources (OERs) such as tutorial videos, and a scenario in which the ultimate goal is to help learners to learn by recommending relevant and high-quality content that matches the students' needs. In order to help learners to learn optimally, knowing *what* the learner needs and providing the *right* content that suits them is crucial. For the former (*what* is needed), we can ask for the student for a specific math problem they want to understand better; alternatively, we could estimate automatically the most beneficial content by analyzing their performance on prior examinations (e.g., if a student does not know much about topic i but they know a lot about topic j, then a video about topic i is probably more useful for them than a video on topic j). For the latter (*right* content), a current challenge with contemporary OER repositories is that the content within each resource is typically poorly annotated, with tags that are too general, e.g., "algebra" or "linear equations" rather than "Simplify $\log_{10} 1000$.". Given the high labor and time involved in manual annotation, it is desirable to devise methods of *automatically* analyzing OER content and devising representations that can facilitate efficient search and categorization.

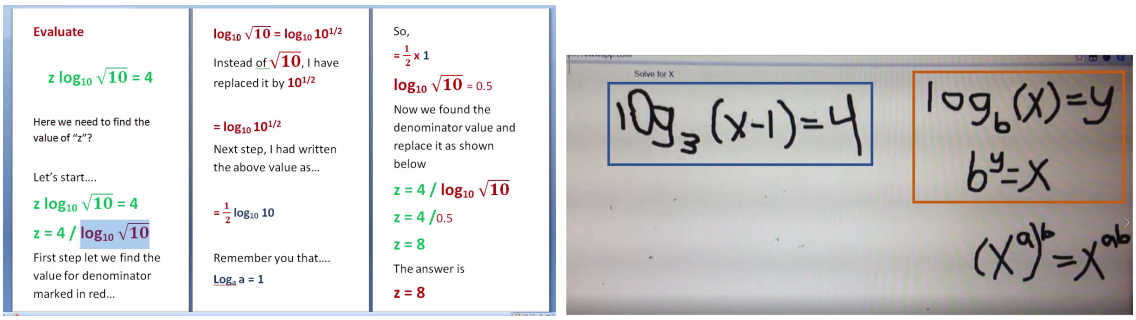

Figure 1.1: Two example videos in our study. For the bottom video, Google's Speech-to-Text API extracts the text "and we're going to solve for x ok our problem is log base 3 of x minus 1 equals 4 we're good then log base B of X is equal to Y and this is equivalent to B to the Y equals X", which shows high agreement with the visual content in the video.

To characterize math videos for their content, one could consider both visual (what is shown) and auditory (what is said) representations. For the former, while optimal character recognition and handwriting recognition are both mature fields, they are typically evaluated in much more constrained settings than math tutorials, in which math is mixed with natural language, and extraneous lines and other graphics can exist (see Figure 1.1). For example, PhotoMath, a population smartphone application that uses the phone's camera to recognize and solve mathematical equations, requires that the math expression be approximately pre-segmented. In full-fledged tutorial videos, this segmentation can be very challenging.

Our research focuses instead on analyzing the speech transcript of the video (while ignoring other potential audio characteristics such as background noise, pitch, etc.). When a particular expression or equation is presented in a video, there is a high chance that the speaker will also say that expression/equation out-loud to the learners (Figure 1.1). Rather than manually transcribing the text from the video, we consider only fully automatic approaches based on automatic speech recognition (ASR; we used the Google Speech-to-Text API in our work). Hence, the text representations we explore must contend with imperfect transcripts. We then

assess the utility of the proposed representations for three tasks: (1) *cluster* the videos automatically into the specific math problems that they explain; (2) *search* through a library of videos for one that explains a particular math problem; and (3) *predict* the individual learning gains of students who watch the videos in a pretest/treatment/posttest paradigm. In these ways, we hope to make available to students the *right* content that is already available, but not easily findable, among large-scale OER repositories.

We conduct our investigation on a collection [1] of math tutorial videos about logarithms, and another dataset from YouTube on basic algebra. Our goal is not just to make coarse distinctions between videos about "algebra" versus "geometry", but rather fine-grained distinctions about specific math problems. Mirroring our goals from the previous paragraph, our research questions are the following:

RQ1: How accurately can the devised text features cluster math tutorial videos into *fine-grained* categories about the specific problem they are solving, and which aspects of these representations are most important?

RQ2: By how much can we reduce the search time of a user who wants to find a relevant video?

RQ3: Are the text features predictive of the individual learning gains of students who watch these videos in a pretest/posttest setting?

Chapter 2

Background

2.1 Text Representations

There are several prominent text representations used for language modeling: (1) Term frequency and Inverse document frequency (TF-IDF) [2]: It characterizes each of a set of documents by the relative frequencies of a finite set of tokens contained within them. TF-IDF has several variants depending on the functions used to compute the TF (e.g., raw count, binarized count) and IDF components (e.g., log). TF-IDF features typically do not require training and are thus suitable for unsupervised settings. The representation is usually easy and efficient to compute, and it lends itself to bag-of-words models for downstream classification. (2) Word embedding models [3], [4]: These models are based on neural networks that are trained using supervised learning; they map each token into a real-valued vector whose location in an embedding space carries semantic meaning. (3) Sentence-level language models such as BERT [3]: These are also based on neural networks, but in contrast to the first two feature types, they operate on whole sentences and can thus capture higher-order semantics.

2.2 Video Content Categorization & Clustering

For categorizing video content automatically, much of the prior work has focused on classifying films into their high-level genres [5], [6] rather than highly specific content like we pursue. One example of classifying videos into narrower sub-categories is by [7], who focus on sports videos. Most prior methods on video categorization focus on visual aspects such as frame transitions, object detection and segmentation. Some as them include the audio (e.g. [5]) such as the audio frequency and amplitude statistics as an additional feature. We are unaware of any previous research that clustered video content at the low-level tags of individual math problems.

2.3 Video Retrieval

As OER repositories have grown dramatically during the past 10 years, there has been increasing interest in the task of video retrieval. Many works in this area have pursued combined feature representations with both textual and visual information [8]–[10]. Yang et al. [9] proposed a method combining Optical Character Recognition (OCR) and ASR to help learners search for specific keyword in German lectures. Hürst [10] found that the lecture slides are more useful than the corrected transcriptions. In our work, while we focus solely on text representations that are automatically extracted using ASR, the features we devise could be easily combined with visual features.

2.4 Estimating the Effectiveness of OERs

For the task of estimating the effectiveness (e.g., associated learning gains) of viewing tutorial videos, researchers have pursued various approaches, including estimating

their effectiveness through correlated measures such as engagement while watching the video [11]–[13]. For estimating the effectiveness of OERs in general, one can also use a combined experimental and reinforcement learning-based approach such as bandit algorithms [14]. While Rafferty et al. [14], [15] suggested the potential use of *context* (for example, features of the OERs as well as of the students' prior knowledge) for predicting learning gains, they did not actually pursue that approach.

Chapter 3

Text Representations

In this paper we explore unsupervised representations of the transcripts of math tutorial videos. When designing the representations, we considered the following characteristics: (1) Similar content should involve similar tokens. A math video whose transcript consists of just "two plus three", for example, is unlikely to be similar to a video whose transcript is "four times x". (2) The most important tokens tend to recur within a video transcript. Conversely, tokens that are uttered only once are often less important or even be transcription errors. (3) The relative order of nearby tokens is important for deciphering the math content. For example, "four over two" and "two over four" are different fractions, but the difference is reflected only in the relative order of tokens, not in their frequencies. For characteristics (1) and (2) above, we created several variations of "1D" text representations that capture which tokens occur more frequently in each video. With the additional characteristic (3), we also explored "2D" text representations that can capture the relative order within a fixed radius from token i w.r.t. token j for each (i, j) pair.

The straightforward way to extract text features that represent the tutorial videos' content is to find the sequence of words that constitute each math equa-

tion or expression. For example, in Figure 1.1 (right), the text in the blue box "log base 3 of x minus 1 equals 4" will be considered as one expression. Nevertheless, we note that extracting the precise mathematical expression or equation from the transcript is inherently ambiguous. For example, the two distinct expressions 2^{x+2} and $2^x + 2$ would likely both be spoken as "two to the x plus two", which is indistinguishable without the visual perception. Fortunately, our objective is not to capture the math content perfectly, but to capture enough of it to enable effective clustering, search, and prediction of learning gains.

3.1 Speech-to-Text Transcription

All the feature types we explore are based on obtaining an approximate transcript of the video from an ASR. In particular, we use Google Speech-to-Text API. As a pilot test of its accuracy on the OERs in our dataset, we manually annotated 10 videos (in total of 3044 words in the ground-truth transcripts) and compared it to the ASR. Google's API achieved a word error rate (WER) of 5%, which intuitively seemed sufficient for our purposes. An example of extracted speech is shown in Figure 1.1 (caption). After obtaining the transcript for each video in our collection, we then tokenized it and summarized the token frequencies.

3.2 Token Types

3.2.1 Individual Token

As our simplest representation, we call each word (separated by space) a *token*, and then we count the number of math-related tokens. For example, if the transcript contains "hello everyone today we will solve the equation log of 5 base 10", it

gets tokenized to the sequence ('hello', 'everyone', 'today', 'we', 'will', 'solve', 'the', 'equation', 'log', 'of', '5', 'base', '10'). Next, we removed all token that are not math-related, defined as: (1) numbers (digit-only), (2) operations (e.g. $+, -, \times$), or (3) variables (an alphabet). For the operations, we map synonyms to the same token, e.g., 'plus' to '+', 'to the [power]' to '^'. Additionally, we add the words corresponding to each digit 0 to 9 (i.e. 'zero', ..., 'nine') as math-related tokens. For variables, we used a restricted alphabet consisting of $\{b, c, n, m, w, x, y, z\}$ (we omitted 'a' since it is also a common English word), which we found worked better than $a - z$.

3.2.2 Expression Token

To infer which math problem a video presents, it might be useful to extract the entire equation or expression, not only the variables and numbers. For example, "2 plus 3" could be considered as one token '2+3' not '2', '+', and '3'. We simply concatenated the (adjacent) sequence of math-related tokens together in the alternate fashion: literal, operator, literal, operator and so on. So, we had the text representation where each index is the frequency of occurrences of each expression.

Specifically, this feature is extracted as follows. (1) We mark all tokens in the transcript as either math-related or non-math-related. Tokens that are labeled as math-related are literals (LIT) and operators (OP) in mathematics such as plus (+), minus (−), square root, log, etc. (2) For each consecutive math-related token sequence, we read each token one-by-one and concatenate each token into one math expression by this specific rule: the math expression must start with LIT followed by OP, LIT, OP, and so on (alternately). If, at some specific location where this rule fails, we end and store the previous expression into our dictionary and start over with the "new" expression parsing process continuing from where we are. For

example, if we have a text "x plus 2 equals 4 and $6 - y = 5$", the total of 2 expression tokens will be tokenized: '$x + 2 = 4$' and '$6 - y = 5$' because the rule fails when encounter the word 'and' so it wraps up the previous expression up to that point as one expression token and starts to look for the new expression. Not that it cannot capture some math expressions if there exists random (non-math-related) tokens interrupt in between the literals and operators.

3.3 Token Count Vector

Given the sequence of tokens in each video, we then compute either a 1D vector or 2D matrix of frequency statistics (which are finally summarized as described in Section 3.4. In the subsections below we let $\mathcal{T}$ be the set of all tokens that appear in any of the videos.

3.3.1 1D (No Order Dependencies)

The count vector of each video contains $|\mathcal{T}|$ components, each of which records how many times the corresponding token occurs in the video.

3.3.2 2D (First-Order Dependencies)

With the goal of encoding the *relative order* of tokens in the transcript, we also tried a method based on computing a 2D *matrix* M, of size $|\mathcal{T}| \times |\mathcal{T}|$, such that M_{ij} is the number of times that token i (if any) appears before token j (if any) in the transcript. In this approach, we introduced a "radius" parameter k to limit the distance of token pairs (i, j) that need to be considered. For example, if $k = 4$, all token pairs (i, j) such that the distance between i and j is ≤ 4 will be counted, otherwise, ignored.

Compared to the 1D approach, the 2D method is more powerful since it can capture order dependencies. While there is a greater risk of overfitting (since there are $|\mathcal{T}|^2$ features), it may offer an advantage on larger datasets.

3.4 Token Summarization Methods

Given the token count vector computed in Section 3.3, we then summarize each count x using a summarization function f. We considered the following functions:

- **Raw Frequencies**: In the simplest implementation, we let $f(x) = x$.
- **Binarized Frequencies**: Binarizing the counts x might be less susceptible to noise; hence, we tried setting: $f(x) = 1$ if $x \geq 1$ and $f(x) = 0$ if $x = 0$.
- **Weighted Frequencies**: It might be beneficial to weight down tokens which appears only once because tokens that appear only once during the extraction process in each video might be noise. In other words, if the token appears only *once* (we call it $t_{=1}$) in a video, its weight will be weighted lower than tokens that appear *more than* once ($t_{>1}$). Intuitively, important tokens should be mentioned multiple times in the video; token found only once are either insignificant or incorrectly extracted. Instead of removing $t_{=1}$, we introduced the parameter r to downweight $t_{=1}$. In this case, instead of having the raw frequencies, We fixed the weight of $t_{>1}$ as 1; however, we downweight $t_{=1}$ by r (e.g. if $r = 2$, the weight of $t_{=1}$ will be $1/2 = 0.5$). We thus let $f(x) = 1/r$ if $x = 1$, $f(x) = 0$ if $x = 0$, and $f(x) = 1$ if $x > 1$. Note that when $r = 1$, this can be seen as Binarized counts.

3.5 Fixed-Size Representation (Dataset Independent)

Due to the potential of instability in vector representation size of the dataset dependent representation (previous methods), we introduce data independent representation (i.e. it would not be affected by the dataset in terms of vector representation size). The high-level idea is that we will consider the numbers and letters 'a' to 'z' as tokens. We then count the frequency for each variable, but for the numbers, since numbers are the infinite set we will look at each digit (0 to 9) in each location (e.g. ones, tens) instead. Note that the representation also depends on the maximum number of digits of the numbers. But we can enlarge the digit dimension by a little amount to capture more digits if needed, e.g., increasing the dimension from 10 to 11 can capture all 11-digit numbers.

The algorithm works as follows: Let any integer n that occurs in the video be represented as $\sum_{i=0}^{k} c(i)10^i$, where c gives the coefficient for the i-th power of 10. We can then represent this number with a sparse matrix A that has a value of 1 at each location $(c(i), i),\ \forall 1 \leq i \leq k$. We then flatten this matrix into a $10 * k$-element vector. Note that this representation has some weaknesses that it cannot distinguish the difference in some scenarios such as between having 23 and 45, and having 25 and 43. The representations of both scenarios would be exactly the same.

Chapter 4

Dataset

We applied the text representations above to two sets of math tutorial videos: (1) *Logarithms* and (2) *Algebra*.

Logarithms: This is the dataset collected by Whitehill & Seltzer [1], which contains both a repository of 208 math tutorial videos about logarithms. Most videos are between 1-3 minutes long. In total the collection spans 18 logarithm problems, with 9 to 17 videos per problem. Relevant only to Section 7, the dataset also contains students' pretest and posttest scores of 541 participants from Amazon Mechanical Turk who watched the videos. There are 226 males, 207 females, and 108 of undefined, with the average age of 33.71 $\pm$9.84. Specifically, each participants were asked to answer 19 logarithm pretest problems, which was classified into 3 main categories: (1) the logarithmic term without variables e.g. $\log_9 1$, (2) the logarithmic term with variables e.g. $\log_w \frac{1}{w}$, and (3) the logarithmic equation e.g. solve for x where $x \log_4 16 = 3$ (category 1, 2 and 3 contain 102, 61, and 45 videos, respectively). Then, they were assigned to *one* random video among 208 logarithm tutorial videos, and were asked to complete a posttest (same level of difficulty as the pretest but slightly different problems).

Algebra: For the search task, we collected another dataset, containing 234 algebra math tutorials on Youtube (because we need many *different* math problems for the retrieval task). As of 234 videos, 213 of them contains *one* math problem and 21 of them contains *multiple* math problems (total of 87 math equations). We manually annotated which equation (e.g. $2x^2 - 2x - 12 = 0$, $x + 7 = 10$) each video explains. For videos that contain multiple math problems, we marked the start end time of each problem.

Chapter 5

Clustering: Video Categorization

Given the different feature types described above, we test whether they serve as an effective basis for clustering the videos. In this section, as ground-truth cluster labels, we took the math problem (there were $K = 18$ unique problems in total) that each video explained as its label. Note, however, that we could also cluster the videos by the *category* of problems that they explain (see Chapter 4); we do so in Chapter 7.

Methods: For each of the different text representations, we applied K-means clustering to group the videos into $K = 18$ clusters, followed by the Hungarian algorithm [16] to optimally match from the estimated cluster indices to the ground-truth indices. Since K-means converges to different local minima depending on the random initialization, we executed the algorithm 512 times and then applied one of two alternative methods for evaluation: (1) We computed the accuracy for the clustering with lowest sum of squared distances; and (2) we computed the average over all 512 trials. As a baseline, we considered a method (averaged over 512 simulations) that "knows" the size of each cluster according to the ground-truth labels and then randomly assigns videos to the clusters.

Results: Table 5.1 shows the clustering accuracy results. We found that the clustering with lowest associated sum of squared errors (SSE) typically gave better accuracy than an average of random local minimum chosen from a set of 512 centroids. We thus report the *accuracy of best local minima* alone. All three methods yield accuracies that are much greater than the random baseline, which achieved only 18.27% accuracy.

5.1 Feature Modifications

5.1.1 Binarizing Token Counts

The results improve significantly on the *frequency counts* for all methods (column 2 in Table 5.1), suggesting that the binary representation is more robust against overfitting.

5.1.2 Weighted Frequencies

We tried multiple values of r, e.g., $r = 2, 4, 8$. Note that $r = 1$ is equivalent to the Binarized Token Counts. We also added $r = 0.5, 0.25$; this contrasts with our intuition for when it weights $t_{=1}$ more; we added this as a sanity check that the accuracy should be getting worse. Table 5.1 shows that the weighted frequencies increase the accuracy significantly up by 10% on average. $r = 2$ performs the best among $r = 2, 4, 8$. As r gets larger, we see a slight decrease in accuracy. When $r = 0.5, 0.25$, the accuracy decreases markedly.

Table 5.1: The Clustering Percent Accuracy (%)

Math Variables	Unrestricted: a - z		Restricted: b, c, n, m, w, x, y, z			
Methods	Simple	Binarized	Weighted (r)			TF log norm
			0.5	1	2	
Individual (3.2.1)	48.56	63.46	42.79	63.94	68.28	73.56
Expression (3.2.2)	53.85	68.75	50.48	75.00	**83.65**	78.67
Fixed-Size (3.5)	45.19	65.87	51.92	65.87	70.67	71.63
Column	1	2		4	5	

5.1.3 Restricting the Alphabet

For all three tokenization methods, using a restricted alphabet delivered accuracy that was at least as good as with the whole set of math tokens; compare the $r = 1$ column under "Weighted" under the restricted alphabet columns (column 4), to the "Binarized" column under the unrestricted alphabet (column 2).

5.1.4 1D vs. 2D

Table 5.2 shows clustering accuracy with the 2D approach. For the radius $k = 2$ on the Expression token (and using weighted frequencies with $r = 2$), the accuracy increases around 2% compared to with 1D. However, we can see lots of variance in the accuracy over the different k, and hence the advantage may not be statistically reliable.

Table 5.2: The Clustering Percent Accuracy (%) comparing 1D v.s. 2D Representations

Methods	**Dimension**	**Weighted (r = 2)**
Equation Token	1D	83.65
	2D ($k = 1$)	83.65
	2D ($k = 2$)	**85.58**
	2D ($k = 4$)	75.96
	2D ($k = 8$)	71.63
	2D ($k = 16$)	73.56
	2D ($k = \infty$)	51.44

5.2 Comparison to TF/IDF

Our 1D methods of building text representation (i.e., token summarization methods) can be seen as variations of TF-IDF, where only the TF term $f(x)$ is used; in other words, we used a constant 1 for the IDF term. (We experimented with several IDF functions but found that they all worked worse than just 1.) The weighted frequency scheme we tried can be seen as a coarse (piecewise-constant) approximation to the (smooth) log function commonly used as the TF function in TF-IDF. Using TF-IDF (with log for TF and 1 for IDF) and Expression Tokens, the clustering achieved 78.67% for the Expression Token (down about 5% from our *weighted* frequency method). For the Individual Tokens, it performed similarily in accuracy compared to the weighted frequency methods. (See the "log" column in Table 5.1.) In summary, the results provide some evidence that our text representations may yield a worthwhile accuracy advantage over TF-IDF.

Chapter 6

Searching

In this section we explore to what extent the text representations in Chapter 3 can enable a more efficient search through a large collection of tutorial videos for a specific math problem. This could help to reduce the amount of time that students need to filter through the millions of math YouTube videos to find the ones that are most helpful. In pilot experiments, we found that contemporary video search engines, even Youtube, do not consider the transcript when returning search results. They mostly rely on the title, description, tags that authors of the video provided, but not the content itself. Thus, it is an interesting question whether the proposed feature representations might reduce a user's search time to find the desired math problem. Here, we describe two experiments in which we used text representations to help the search task to reduce (1) the number of videos to watch, and (2) the amount of time to watch.

6.1 Architecture

Using the text representations, we can build a simple search engine as follows: (1) From each video i in a collection $\mathcal{S}$, we transcribe its speech into text (using Google

Table 6.1: The Percent Decrease in Number of Videos Watched

Methods	**Dimension**	**Simple**	**Binarized**	**Weighted (r = 2)**
Individual Token	1D	80.08	83.94	83.44
	2D ($k = 1$)	80.80	85.61	84.20
	2D ($k = 2$)	81.28	87.83	86.14
	2D ($k = 4$)	80.49	86.73	87.79
	2D ($k = 8$)	79.55	**88.25**	87.17
	2D ($k = 16$)	79.20	87.95	87.95
	2D ($k = \infty$)	77.80	84.82	84.29
Equation Token	1D	77.91	81.08	81.62
	2D ($k = 1$)	77.66	80.63	81.25
	2D ($k = 2$)	78.19	81.41	81.70
	2D ($k = 4$)	78.44	82.89	82.78
	2D ($k = 8$)	77.81	**83.73**	83.12
	2D ($k = 16$)	76.38	83.48	83.00
	2D ($k = \infty$)	73.59	81.15	80.93

ASR) and then extract its text representation v_i. Then, (2) for any search query (e.g., "Simplify: $\log_4 16$"), we likewise extract its text representation q using any of the methods presented in Section 3. Finally, (3) we rank all the videos in $\mathcal{S}$ by the *cosine similarity* between v_i and q.

6.2 Number of Videos Watched

Here we assume that each video in $\mathcal{S}$ explains a single math problem. To make the task interesting, assume that these problems are all part of a subdomain such as "algebra" or "logarithms"; this focuses the search task on fine-grained (e.g., "$log_2 8$"

versus "$log_8 2$") rather than coarse-grained (algebra versus statistics) distinctions. If the user wishes to find *any* video that explains a specific problem, then by just randomly watching the videos one by one (which, after performing a keyword search, is often how this is done), the expected number of videos they must watch is $|\mathcal{S}|/2$. How much does the search engine help to *reduce* the number of videos the user must watch compared to this baseline? Due to page limits, we only present our main findings.

Methods: The pipeline to test and evaluate is as follows: (1) For each algebra equation we found in the video i from our dataset (see above), we turn them to a search query, say q_i. (2) We extract the text representations (using all methods presented in *clustering* problem, Section 5) from all t's and q's: $t_1, \ldots, t_{213}, q_1, \ldots, q_{213}$. (3) For each q_i, we ranked t's by the highest of *cosine similarity* between their vectors to the query vector q_i, (4) then calculated the number of videos to watch (if we watch them one by one from the list of t's) before the correct one was found, say v_i. (5) We finally averaged v's to be the final $E[\#\ of\ video\ to\ watch]$ and compared with the random baseline ($213/2 = 106.5$ videos), which can be derived mathematically as number of videos divided by 2.

Results on the Algebra dataset: On the 213 videos of the Algebra dataset that explain exactly one problem, the search engine using the 2D Binarized Individual Token ($k = 8$) can decrease the number of videos (compared to the baseline) to watch by 88.25% (Table 6.1). The trends were that the 2D representations were generally more effective, with a higher percentage decrease in search time for nearly all the values of k that we tried. After $k > 8$, the performance started to drop. For the 1D approach, the best text representation was TF-IDF (with log for TF and identity for IDF); the reduction was slightly lower at 86.19%.

Results on the Logarithm dataset: For each of 18 logarithm problems, we

search for *any* of the videos that solve that particular problem. Comparing the results with *random* baseline, the results show the same trend as for the Algebra dataset: The 2D Representation gives the best results. We found, for instance, Binarized Individual Token yields the results of 89.96%, and 93.19% for 1D and 2D ($k = 8$), respectively. The same holds true for Weighted Expression Token ($r = 2$) with the results of 91.25% and 93.20%. For the 1D approach, the best text representation was TF-IDF (with log for TF and identity for IDF); the reduction was slightly lower at 92.85%.

6.3 Amount of Time Watched

Here we consider a setting in which *multiple* math problems may be explained in a single video. A search engine that can pinpoint which *segment* of a video explains the solution could save the user significant time compared to watching the whole video. For this setting, there is a trade-off between granularity and accuracy: the search engine may be more accurate if the segment length is longer, but the user can save more time if the segment returned to them by the search engine is shorter. Hence, we introduced a segment length parameter, L. We divided each video into multiple segments of length L (e.g. 15 secs, 30 secs, 1 mins, 2 mins, etc.). Each segment has its own (sub-)transcript and its own problem that it explains. We can thus conceptualize the task as similar to Section 6.2, but we treat each segment as its own "video". Our goal is to find *any* segment in the video that explains the problem in the user's query q. As a baseline, we used a simulation (repeated 20 times and averaged) to estimate the sum of the segment lengths (in seconds) that a user would have to watch before finding a segment that explains the desired problem.

Results on the Algebra dataset: We analyzed the 21 videos of the Algebra

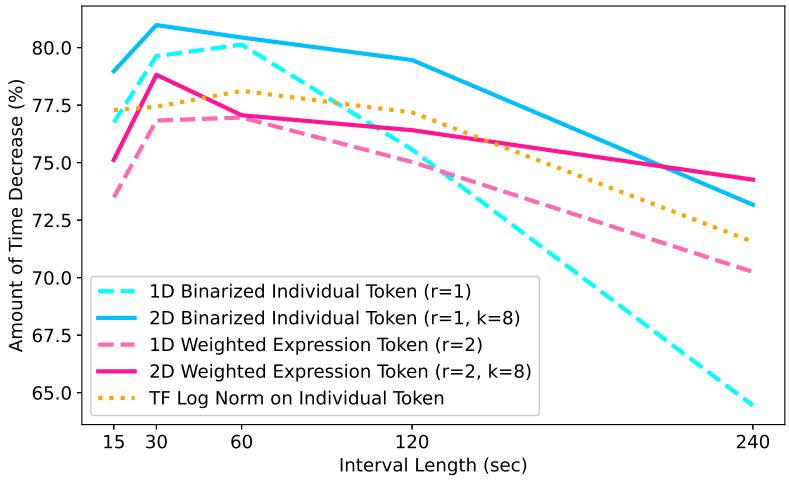

Figure 6.1: The decrease in time needed to find specific math content in a set of math tutorial videos. Each line shows a different text representation over different segment lengths used for evaluation.

dataset that contain multiple problems; in total, these videos explain 87 algebra problems. We varied the segment length L over the set {15s, 30s, 1m, 2m, 4m} (see Table 6.1). The results are consistent with Section 6.2, where the best text representations were 2D Binarized Individual Token ($k = 8$). In particular, the 2D representations showed an advantage (compare the pairs of {blue, pink}'s *solid* and *dashed* lines). We found that radius $k = 8$ for 2D Representation preforms best across each method. For the Interval Length, the percent decrease, at $L \in \{30s, 1m\}$, in watch time is highest (i.e., the most helpful, see Figure 6.1). As L continues to grow, the results go down and at $L = 15s$, the performance seems to drop. This exemplifies the trade-off between (1) the segment length and (2) the available information. For example, the segment length of 15 seconds might be more likely to cut an equation into two parts so that it cannot be recognized.

Chapter 7

Learning gain prediction

In the previous sections, we showed that the proposed text representations can capture the video contents and thereby help to reduce the search time and cluster similar videos together. In this section, we investigate whether the text representation can be used to predict the learning gain of students who watch the videos as an educational intervention. The high-level idea is that the effectiveness of each tutorial video can be estimated by the *interaction* of the *content* within the video and the student's prior knowledge. In contrast to some prior work that predicted the *average* learning gains of a video over many students, here we tackle the arguably harder problem of predicting *individual learning gains* of each student, measured as the difference in test scores on the curriculum before and after watching the video. Our goal is thus to investigate whether the text representations facilitate *personalized learning*.

As described in Section 4, the Logarithms dataset contains pretest and posttest scores of students who received one of the logarithm tutorial videos as an intervention. Hence, we can use each participant's pretest score, as well as the text representation of the video they watched, as predictors to estimate their learning

gains, i.e., posttest minus pretest score. Rather than use the text representation as a feature vector itself, we instead use the *category label* assigned to the problem (Chapter 5) by the clustering algorithm as a 0-1 indicator variable with an associated model coefficient; hence, our models can potentially find interactions between gaps in a student's prior knowledge and the topic that was explained in the video they received.

7.1 Simple Linear Models

Let $p_{ij}, j = 1, 2, 3$, be student i's prior knowledge (pretest score) within the 3 problem categories (j) on logarithms. Let $c_{ij}, j = 1, 2, 3$, be 0-1 indicator variables that reflect whether student i's assigned video belongs to each category j. (Note that each video is assigned to exactly one of the three categories.) We can compute c_{ij} using either (a) Manually Labeled Categories (MLC) from human annotators, or (b) Automatically Labeled Categories (ALC) from the text representations and clustering algorithm (Section 5).

Prediction Model 1: We first consider a linear prediction model in which the individual learning gains (posttest minus pretest) y_i is estimated by $y_i = \sum_{j=1}^{3} (w_j p_{ij}) + \epsilon_i$, where $w_j, j = 1, 2, 3$ are model coefficients and ϵ_i is a sample from a 0-mean Gaussian distribution. We then used an ANOVA to assess the predictive value of the model.

Results: The prior knowledge has a statistically significant effect on the learning gain ($F_{1,\,593} = 10.08$, $p = 1.754e-06$), with the Root Mean Square Error (RMSE) of 0.479.

Prediction Model 2: Based on the results above, we constructed a second model that considers multiplicative interactions between the student's prior knowl-

edge p_{ij} in each problem category and the cluster label c_{ij} of the student's assigned video: $y_i = \sum_{j=1}^{3} (w_j p_{ij} + v_j c_{ij} + u_j (p_{ij} \times c_{ij})) + \epsilon_i$. Importantly, this model contains multiplicative interaction terms $p_{ij} \times c_{ij}$.

Results: We found that the interaction $p_{ij} \times c_{ij}$ using MLC has a statistically significant effect on the learning gain ($F_{11,\,582} = 5.839, p = 5.11e-09$), and so does this interaction using ALC ($F_{11,\,582} = 6.425, p = 4.125e-10$). The RMSE is 0.464, which is slightly better (about 3.1% relative decrease) compared to prediction model 1. Specifically, we found that, for example, u_3 is *negative* and statistically significant ($p = 0.0005$) in the ALC model. The negativity of u_3 means that, if $p_{i3} \times c_{i3}$ is low, then the learning gain is high (and vice versa). In turn, $p_{i3} \times c_{i3}$ is low either because (1) p_{i3} is low and $c_{i3} = 1$, i.e., an individual knows little about topic 3 and receives a tutorial about topic 3, yielding high learning gain; or (2) p_{i3} is high and $c_{i3} = 0$, i.e., an individual already understands topic 3 and receives on another (more helpful) topic, yielding high learning gain.

The fact that both the MLC and ALC interactions were statistically significant (albeit with only a small decrease in RMSE) is evidence that the text representations can group videos in ways that predict their effectiveness for individual learners, and that these text representations might serve as a useful context vector in bandit algorithms [14].

7.2 Deeper Models

For the deeper model, we split up the data into a training and a testing portion, and performed double student-wise cross-validation on the training portion to train models, and then applied it to the testing portion. However, the results did not suggest any consistent benefit of the deeper models compared to the simple linear

ones. We describe the experiments and results in the following sections.

7.2.1 Model Architectures

As our goal is to analyze the effects of the feature and model types in predicting learning gain, we build 3 simple neural network models: linear regression (LR), 1-hidden-layer neural network (1NN) and 2-hidden-layer neural network (2NN). We build the 1NN and 2NN with the following architecture: Input layer followed by hidden layers using the ReLu activation. On each hidden layer, it is followed by the dropout layer and finally the output layer is a dense layer maps to 1 final value which is the normalized learning gain. We left the number of neurons in each hidden layer and the dropout percentage as the hyperparameters at training time. We use the mean squared error loss as our evaluation metrics

7.2.2 Implementation

We use double cross-validation techniques to run the model on 5 folds to benefit from all the data. The inner cross-validation is for tuning the hyperparameter to find the best hyperparameter, says h*, on each fold. While the outer cross-validation will train the model on all training data each fold (i.e. 80% of entire set of data) using h* as the hyperparameter and test on 20% left. We split the data (fix the seed so that every time we run we get the same splits) using two scenarios described before: (1) predicting learning gain (of new students) from existing videos and (2) predicting learning gain (of new students) from new videos. We will split the data into 5 folds randomly in scenario 1. The same is applied to scenario 2, but we will make sure that the sets of videos in training and testing set are disjoint. This means that the videos in the test set have never been trained on before, which represents the new video term we mentioned in scenario 2. Finally, we average the correlations

of the predicted learning gain versus the ground truth among all 5 folds.

Our hyperparameters are (1) the neural unit in each hidden layer: $\{16, 32\}$, (2) batch size: $\{4, 32\}$, (3) dropout percent: $\{0, 0.1, 0.5\}$, and (4) numbers of epoch: $\{100, 250\}$.

7.2.3 Experiments and Results

To assess how differences between the model types and feature types affect the accuracy of predicting subjects' learning gains, we conducted a randomized experiment on Mechanical Turk. Each participant was asked to do a similar survey as conducted in [1] by Whitehill Seltzer, except some irrelevant data such as their demographic information. Thus, we make a slight change to the survey template and collect the total of 100 data points. The steps of analysis will be done as follows: (1) running the 5-fold double-cross validation on the training data, (2) given the results from training the models, we will make hypotheses on the results we found, (3) we then test the hypotheses on a completely new dataset collecting from MTurk.

Do model types and feature types affect the predicted learning gain? (RQ3) We will focus on two main questions (1) Effects of Model Types and (2) Effects of Feature Types by including the results of 5-fold cross-validation (training state), the hypotheses we make after seeing the trained results, and test results on new MTurk collected dataset.

First, in order to tell whether the model types (LR, 1NN, and 2NN) affect the prediction task, we run the 3 models on different feature types to see the differences between using linear regression and neural networks. The results are shown in Table 7.1. The values are the correlation of average predicted normalized learning gain and the ground-truth. We run each model (i.e. each cell in the table) 10 times so get 10 model predictions, then we ensemble them by averaging to get the final prediction

Table 7.1: Existing Video Correlations

	Training			Testing		
Inputs	**LR**	**1NN**	**2NN**	**LR**	**1NN**	**2NN**
PS	0.06	0.05	0.04	0.56	0.57	0.61
PV	0.29	0.29	0.31	0.21	0.35	0.56
PV+3C	0.30	0.23	0.27	0.20	0.36	0.51
PV+18P	-0.003	0.33	0.32	0.22	0.35	0.54
PV+ER	0.26	0.27	0.23	0.23	0.24	0.24
PV+NVR	0.26	0.20	0.12	0.21	0.16	0.04
PV+C-ER	0.30	0.27	0.25	0.17	0.25	0.43
PV+C-NVR	0.30	0.29	0.30	0.23	0.38	0.55
Average	0.22	0.25	0.23	0.25	0.33	0.44

Table 7.2: Existing Video ANOVA

Variables	**Training (p-value)**	**Testing (p-value)**
Models	0.37	0.09
Features	0.43	0.96

(we run this to minimize the variance of our analysis). The higher correlation is better. The last row shows the average correlation of each model type.

Scenario 1 Training State: We can consider (1) the difference between model types. In the training state, from the average correlation (last row in Table 7.1 on the left side), we do not see any significant changes in the correlation value. Though, in some cases the correlation is very low such as -0.0029 of the PV + 18P with LR model. This can happen just by chance that the data we split (5 folds) does not have a good linear relationship with the learning gain values because other features in the LR model do not have the same trend. Additionally, the ANOVA yields the p-value of 0.38 (>0.05) for the model-type analyses. Therefore, we hypothesis that

H1: The model type does not have any significant effects on the learning gain prediction

For (2) the difference between feature types, the correlation of other features we add to the baseline PV does not seem to have a significant effect on the learning gain prediction. As the PV correlations are ranging around 0.29 - 0.31 which is the same for other feature cases. Some features lead to a significantly lower correlation such as -0.0029 of PV + 18P with LR model. This might suggest that the input features PV + 18P does not have a good linear relationship with the learning gain values. But this does not mean that PV + 18P is a bad feature since, in the 1NN and 2NN models, PV + 18P performs slightly better than other models. There are two interesting results here (i) the PV + NVR in the 1NN and 2NN model performs worse than other models (except the PS that we will discuss next) which yields lower than 0.20 in correlation values. This might be because of the overfitting training data (80%) problem. Although the ANOVA results show that the feature types does

not have a significant effect on the learning gain (p-value of 0.43), in the (training state) results, (ii) it suggests that PV performs much better than the PS. The PS yields very low correlation in all of the model types. Thus, we hypothesis that

H2: The feature type does not have any significant effects on the learning gain prediction, but **H2.1:** The pretest vector (PV) is a better based-feature than pretest score (PS). In other words, the pretest vector gives a much higher correlation.

Scenario 1 Testing State: To test our hypotheses, we collect the new data from MTurk using the same setup. Although the average learning gain suggests that deeper models yield higher correlation, the ANOVA does not support this with the p-value of 0.09. Thus, we conclude that correlation values are different across the model types just by chance and do not have a significant effect on the learning gain results which support our H1. For the feature-type analyses, the range of correlations are wider than in the training state; however, the ANOVA suggests that there is still no significant effect on the learning gain prediction (supports our H2). For H2.1 (the sub-hypothesis about the comparison between PV and PS), we get a very contrasting trend on the testing state. The PS yields much higher correlation than in the training state (contradicts our H2.1). This might be because the variance of the data in the training set and new collected data (though both are from MTurk but different time). To test this, we run a quick analysis on the correlation between the pretest scores and learning gain values. The correlation of these of the training data is 0.0854 whereas this of the testing data is 0.5572. This suggests a huge difference which could mean there is a lot of variance from day to day in the population and/or behavior of MTurkers.

Scenario 2 Training State: In this setup, we do the same analyses and setup

Table 7.3: New Video Correlations

	Training			Testing		
Inputs	**LR**	**1NN**	**2NN**	**LR**	**1NN**	**2NN**
PS	-0.19	-0.39	-0.45	0.57	0.55	0.59
LN	-0.10	-0.06	-0.12	0.03	-0.05	0.03
PV+LN+NVR	0.12	-0.006	-0.16	0.05	0.24	0.38
PV+LN+CER	0.12	-0.05	-0.07	0.12	0.35	0.33
WRD	-0.14	-0.02	-0.12	-0.15	0.14	-0.12
PV+WRD	0.13	-0.06	-0.08	0.15	0.41	0.30
PV+WRD+NVR	0.11	-0.05	-0.22	0.06	0.09	0.17
PV+WRD+CER	0.13	-0.05	-0.07	0.12	0.35	0.33
Average	0.04	-0.07	-0.15	0.12	0.26	0.24

Table 7.4: New Video ANOVA

Variables	**Training (p-value)**	**Testing (p-value)**
Models	0.0008	0.000003
Features	0.10	0.08

as in scenario 1. Table 7.3 suggests that the LR might be more suitable to predict the learning gain than neural nets as the average correlation keeps decreasing as the model goes deeper (deeper model is worse). This might suggest the model is unable to predict the learning gain on the new video. In other words, in order to see how good the video can help the learners learn, we might have to have the set of data (e.g. small sample) on that video to train (e.g. the existing video scenario). Since the ANOVA result suggests that the p-value of 0.0008 for the model type, this means that is a significant effect on the model type when it comes to predict the learning gain. Thus, we hypothesis that

H3: The model types have a significant effect on predicting the learning gain (deeper is worse)

For the feature there is no significant high result since most of the correlation results we get is negative. Together with the ANOVA results yields p-value of 0.10 (>0.05), we hypothesis that

H4: The feature type does not have a significant effect on predicting the learning gain.

Scenario 2 Testing State: Table 7.4 (right side) shows the ANOVA analysis results, yielding the p-value for model-type independent factor of 0.000003 suggesting that model type brings a significant effect on the learning gain prediction. However, Table 2 (right side) shows the significant improvement using neural nets over the linear regression (the deeper is better on average) but in the training state (left) it suggests the deeper is worse. Thus, we can roughly conclude that the model types

do affect the learning gain prediction statistically (partially supports H3), but in which way is still unexpectable. This suggests a lot of variance in the data collected from time to time from MTurk and might need further investigation on the effect of the model types. For the feature types, we can conclude as the same in scenario 1 which is there is no significant effect on the predicting learning gain (supports H4). Note that the correlations using PS as based-feature is still high (0.5 approximately) which due to the possibility of the variance in collected data we have discussed in the previous scenario section.

Can we teach students by automatically selecting a video according to a policy? In this section, we run a simple experiment focusing on the question of selecting the (probably) best video to a student based on the pretest history data of the student. Since in the previous section, we run different model policies and get their correlation. We can use these policies to train and predict the learning gain of the new student for a specific given video. Then, we can assign the best video that has the highest predicted learning gain to the student, and hopefully they will learn the most (compared to other videos given). Hence, we conduct another MTurk experiment by collecting 200 new data using the same format survey as we use to answer RQ3 (the experiment is conducted on a different day). However, the differences are that, in this new MTurk experiment, we will not assign a random video, but rather train and provide the best video.

We pick three policies to run: (1) the *best-video* policy which always gives the best video that has the highest average learning gain in the training dataset (the best-video policy gives the same video always). (2) the PV+ID policy (pretest vector and each video unique id). We include this in order that the policy will treat each video independently not like in other previous features we have that always combine the videos into classes (categories). (3) the PV+C-ER policy (pretest

Table 7.5: The Average Learning Gain of Students

Policies	**Average (Normalized) Learning Gain**
Best Video	0.081
PV+ID	-0.029
PV+C-ER	0.035

vector and clustering equation representation), we pick this model because PV+C-ER policy includes (automatically extracted) text feature representations which can be a helpful thing to see whether text representation has some effects.

Given these 3 policies, we modified the MTurk experiment by, at the start of the session, each participant will be randomly assigned to one policy. Then, throughout the experiment, we use this policy to analyse and assign the video to students. Table 7.5 suggests that by training policies using standard supervised learning (not re-sampling in any way), trained policies (e.g. PV+C-ER, PV+ID) do not do better than just giving the best video since the average learning gain of the students who are assigned the video by best-video policy is higher than the average of using the PV-C-ER and PV+ID policy. As our previous results suggest that the feature types have no statistically effect on the learning gain prediction, the variance in the learning gain might be random by chance.

We also compared these results to the new 100 collected in Section 7.2.3. The survey was conducted by randomly assigned the video. We found that the average of the learning gain is much higher (average learning gain = 0.164). This means when everyone gets a random video, the average learning gains are much higher than during the day we collected 200 data points using the fixed policies. Since the different is significantly high and the experiments are run during the different days,

this suggests that there might be a lot of variance from day to day in the population and/or behavior of MTurkers

Chapter 8

Conclusion and Future Work

We have devised novel text representations, based on speech transcripts extracted automatically from an automatic speech recognition service, to represent the content of math tutorial videos. On a dataset of hundreds of math videos and hundreds of students who watched them, we showed that the proposed text feature vectors can be used to (1) accurately (around 85%) cluster the videos into the math problems they solve (RQ1); (2) search for specific video content in a large repository of videos, thereby saving the user considerable (up to 88%) search time (RQ2); and (3) predict individual learning gains, in conjunction with features of the students' prior knowledge, with statistical significance (RQ3).

Specifically, we found some mild evidence that 2D representation (ordering) can be better than 1D, though there is a risk of overfitting. Also, using the text summarization other than the raw count yields consistently better results. For the retrieval task where one video contains many problems, we found that interval range of $30, 60$ seconds are the most beneficial, balancing between the information provided, and the length (time-consumption).

For the clustering problem, we provided evidence that the proposed representa-

tions outperform TF-IDF and its variants. One possible reason is that TF-IDF was designed for natural language, whereas our feature representations were designed from the ground up for math content.

However, we have presented that the model types (linear regression, neural nets with different depth) and the feature types does not have any significant effect on the learning gain prediction. It might do a better job in some cases just by chance. We investigated and tested whether the trained policy can be helpful to assign the best video to the student (hoping that the students will learn the most compared to all other videos); however, we found out that the best video which has the highest average learning gain (from history data) gives the better results. Also, the average learning gain of students is significantly different from days to days (as we conducted 2 MTurk experiments). This could have implications for the open educational resources (OER) that the task of predicting the learner's learning gain might not be obvious using just the pretest and resourced video information. Students also vary from day to day and might have different backgrounds and learning improvement curves that one should consider. However, we have presented that the text-representation of the video content is useful in some settings.

Future work can explore how to use the text representations to serve open educational resources such as tutorial videos to real users in real time, for example, for contextual bandit algorithms. We can also continue to explore whether deeper prediction models can be trained without overfitting.

Bibliography

[1] J. Whitehill and M. Seltzer, "A crowdsourcing approach to collecting tutorial videos–toward personalized learning-at-scale," in *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, 2017, pp. 157–160.

[2] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[5] S. Fischer, R. Lienhart, and W. Effelsberg, "Automatic recognition of film genres," *Technical reports*, vol. 95, 1995.

[6] V. Kobla, D. DeMenthon, and D. Doermann, "Detection of slow-motion replay sequences for identifying sports videos," in *1999 IEEE Third Workshop on Multimedia Signal Processing (Cat. No. 99TH8451)*, IEEE, 1999, pp. 135–140.

[7] E. Sahouria and A. Zakhor, "Content analysis of video using principal components," *IEEE transactions on circuits and systems for video technology*, vol. 9, no. 8, pp. 1290–1298, 1999.

[8] F. Wang, C.-W. Ngo, and T.-C. Pong, "Structuring low-quality videotaped lectures for cross-reference browsing by video text analysis," *Pattern Recognition*, vol. 41, no. 10, pp. 3257–3269, 2008.

[9] H. Yang and C. Meinel, "Content based lecture video retrieval using speech and video text information," *IEEE transactions on learning technologies*, vol. 7, no. 2, pp. 142–154, 2014.

[10] W. Hürst, T. Kreuzer, and M. Wiesenhütter, "A qualitative study towards using large vocabulary automatic speech recognition to index recorded presentations for search and access over the web.," in *ICWI*, Citeseer, 2002, pp. 135–143.

[11] A. Ramesh, D. Goldwasser, B. Huang, H. Daumé III, and L. Getoor, "Learning latent engagement patterns of students in online courses," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 1272–1278.

[12] A. S. Lan, C. G. Brinton, T.-Y. Yang, and M. Chiang, "Behavior-based latent variable model for learner engagement.," *International Educational Data Mining Society*, 2017.

[13] S. Bulathwela, M. Pérez-Ortiz, A. Lipani, E. Yilmaz, and J. Shawe-Taylor, "Predicting engagement in video lectures," *arXiv preprint arXiv:2006.00592*, 2020.

[14] A. N. Rafferty, H. Ying, and J. J. Williams, "Bandit assignment for educational experiments: Benefits to students versus statistical power," in *International Conference on Artificial Intelligence in Education*, Springer, 2018, pp. 286–290.

[15] J. J. Williams, J. Kim, A. Rafferty, S. Maldonado, K. Z. Gajos, W. S. Lasecki, and N. Heffernan, "Axis: Generating explanations at scale with learnersourcing and machine learning," in *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, 2016, pp. 379–388.

[16] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.