Predicting Lecture Video Complexity: Analysis of Supervised Regression

Nick Su Ismael Menjivar
njsu@stanford.edu menjivar@stanford.edu

December 8, 2014

Abstract

In the past decade, use of *Massively Open Online Courses (MOOCs)* has rapidly risen, providing millions of students access to higher education. MOOCs have also changed the education paradigm, with most courses involving a combination of short video lectures with moderated online discussion. This project evaluates performance of different supervised regression algorithms and features to predict lecture video complexity. We find that there is a weak relationship between our predictors and labels, but much promise for future work with more advanced natural language processing (NLP) techniques. We hope the results of this work will help MOOC instructors tailor their teaching style to virtual audiences.

Introduction

The analysis performed by this project is based on data from Stanford's OpenEdX online learning courses. These courses consist of many short videos each around 10 to 20minutes in duration. While the courses cover a broad range of topics, many of them share the same video structure: an instructor visible on the screen speaks in the foreground or corner of the screen while a slideshow is shown in the majority of the screen space. Instructors will often move and handwrite text onto the screen, but the structure is otherwise very uniform.

Our goal was to design a model that provides information to an instructor on where students are having difficulty following a lecture by examining the progression of text visible on the screen and the progression of closed caption text up until that point. This function is constructed using a supervised learning algorithm informed by a dataset of user interactions with the video during different time intervals.

Data

Our project used three data sets that provide analysis on forty 15- to 20-minute course videos for "CS144: Intro to Computer Networks":

1. A dataset from the Vice Provost Office of Online Learning contained almost six-hundred thousand user interactions with the videos when they were offered on OpenEdX.
2. Data of text-on-screen was extracted from frames of each video screen using MATLABs optical character recognition (OCR) tool.
3. Closed caption text files were extracted from each video online using KeepSubs.

Video Interactions Data

The dataset for user interactions with the videos consisted of a single .CSV file with the following information for each of the 594,567 interactions:

- event_type: The type of interaction. This could be pausing the video, playing the video or seeking to a new section of the video.
- video_current_time: The current playhead time. Tells us when a user paused or played a video.
- video_new_time: For video seek, new playhead time.
- video_old_time: For video seek, old playhead time.
- video_code: Machine code name for video. Also video code for Youtube:
  - youtube.com/watch?v=video_code
- anon_screen_name: Unique id for a viewer

OCR On-Screen Text Data

Screen text was manually extracted using optical character recognition at three second intervals in the video. Data is stored in a txt file with lines labeling the video time (in seconds) that OCR was performed followed by a line of all the text identified by MATLAB's built-in OCR function. While it recognizes a majority of text on a given frame, it contains some inconsistencies correctly identifying human writing and squiggles.

Closed Caption Text Data

Closed captions are done manually by OpenEdX and are consequently very clean and error-free. Each caption has the following associated data:

1. Caption Number
2. Time duration that it appears on screen
3. Text displayed

Label Extraction

Labels were extracted from the Video Interaction Data by analyzing the pause and seek event types. We judged these interactions to be most indicative of where a student may be struggling to keep up with the lecture.

For each video, we kept a count of how many interactions occurred at each twenty-second interval. Each pause event at current time *t* increments the count of events during the twenty-second interval in which it took place. For seek type interactions, we first check if a student rewinds the clip (new_time < old_time) and increase the count of interactions for each twenty-second interval within the rewound region.

To normalize the counts, we divide them by the number of users that interacted with the given video to get an estimate of how many times each segment was watched by the average user. These can then be used as a labeling to mark which sections students are pausing and viewing the most, and therefore most likely to be struggling to follow the lecture.

Feature Extraction

Extracting meaningful features proved to be difficult due to the nature of the problem. Unlike typical NLP features that typically operate on bounded datasets and perform classification, obtaining NLP features that reflect time dependence and are suitable for regression is more difficult. In this project, we present and evaluate a set of parameterized base features and evaluate feature strength, parameter sweeps, and alternative feature sets.

Our base feature set contains 14 features, consisting of the frame time, a 9-term set of features drawn from closed captions, and a 4-term set of features drawn from the OCR dataset. Aside from the frame time, the other features are described as follows:

- *Closed Caption Features* examine speech rate and word occurrence and are parameterized on a word depth W. The 9 features are listed as follows
  - *Time Depth* T_d is the time taken by the lecturer to speak the last W words.
  - *Cumulative Term Frequency(CTF)* looks at each word in the last W and examines how often it has occurred prior to time T. The last W words are analyzed, and the $16^{th}/50^{th}/84^{th}$ percentile and Gini index are produced as the four features.
  - *Term Frequency(TF)* generates the same four terms but instead looks at the occurrence of a word throughout the entire duration of the video. The same four features are produced
- *OCR Features* examine text changes from frame to frame and are parameterized on frame depth F.
  - *Average Frame Change* measures the number of new words per frame and generates two averages as features: one over the last F frames, and another over all previous frames.
  - *Average Frame Text* measures the average number of words per frame over the last F as well as all previous frames

In addition to evaluating algorithm performance on the base feature set, three additional tests are made.

1. *Parameter Sweep*: Word and frame depth are swept over a suitable region to determine the optimal depth the feature set should be generated over.
2. *Expanded Feature Set*: An expanded feature set is generated that simply creates N and F size vectors of individual word and frame frequencies. This set is run on PLS and SV regression in the hopes of identifying dependencies between variables not easily quantified in a small feature set.
3. *TF-IDF:* is compared to TF to examine whether inter-data information can improve performance.

Algorithms

Linear Regression

Linear regression characterized different features, feature parameters, and feature strength, providing us with varied correlations within our dataset. This model served as our basis for comparison across the various configurations of cross-validation and feature selection.

Partial Least Squares Regression (PLSR)
PLSR was used to specifically evaluate the size of our feature set, measuring our original 14-term feature vectors against reduced versions and our highly expanded feature vectors. This model was used for reducing feature size of our base feature set and examining whether an expanded set could yield better prediction results.

Vapnik's Support Vector Regression (SVR)
SVR allowed us to see how our other regression algorithms performed against highly optimized solutions. If SVR was able to correctly predict our labels to certain margin, we would hope to replicate the performance in our optimization of variables using other algorithms.

Binary Decision Tree (BDT)
We did not have high hopes for BDT, because our 14-dimensional feature set would naturally lead to over fitting the limited amount of examples we had, but we wanted to confirm our intuitions about the performance of these models.

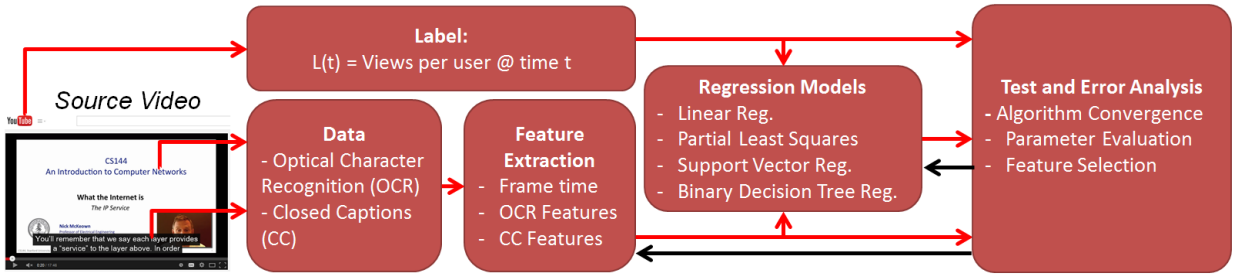
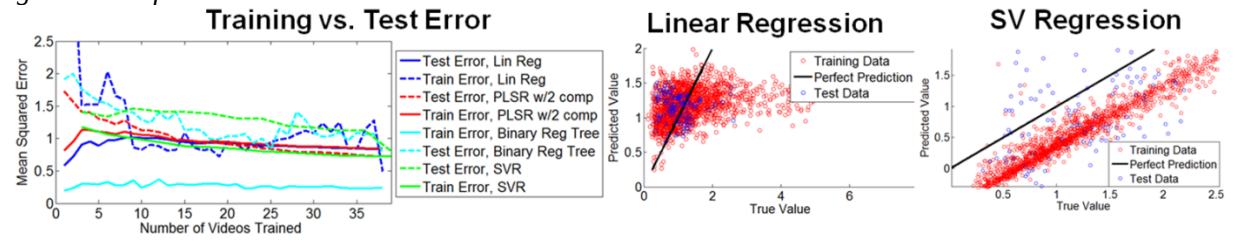


Figure 1: Learning Model Implementation

Figure 1 depicts a high level block diagram the learning implementation used in this project. Tests of parameter and feature strength were based around linear regression, while PLSR was used to evaluate larger feature sets.

Results and Discussion

Algorithm Performance



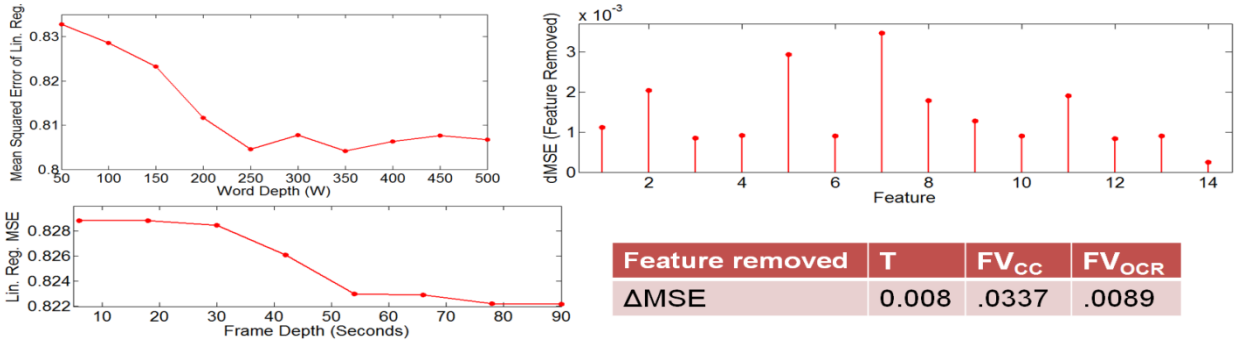
	BDT	Linear Regress	PLSR	SVR
Training MSE	0.236	0.830	0.840	0.720
Testing MSE	1.035	0.867	0.725	0.810

Figure 2: (Top Left) Convergence of test and training error as number of training videos is increased from 1 to the 39 of 40 videos. (Top Right) Scatterplot of observed vs expected values for a training set of 37 videos and a test set of 3 for Lin. Reg. and SVR. (Bottom) Table of convergence for training and test error of Binary Decision Tree, Linear Reg., Partial Least Squares, and Support Vector Reg.

Figure 2 displays a set of summary charts and tables comparing the performance of the four learning models used. Test and training error on the left side figure were obtained using hold-out cross validation on varying numbers of videos. Noise is observed on small hold-out sets, which we have attributed to high levels of feature and label variance between videos, and both PLSR and linear regression have test error below training error as a result. Both linear models converge, while SVR and BDT do not reach convergence.

As seen in the scatterplots on the left, a large amount of variance is observed with both lin reg. and SVR, and a correlation coefficient of .356 is extracted from the linear regression model.

Feature Evaluation



Feature removed	T	FV_{CC}	FV_{OCR}
ΔMSE	0.008	.0337	.0089

Figure 3: (Left) Training MSE as a function of word depth (top) and frame depth (bottom). (Top Right) Stem plot of increase of training MSE with individual features removed. (Bottom Right) Simplified table of training MSE increase with removal of individual feature sets.

Figure 3 presents another set of figures and tables evaluating the strength of different features and parameters in the 14-term base vector that we used. The two plots on the left depict the changes in MSE based on different feature vector parameterizations. As shown on the plots, improvements in MSE level out beyond 250 words and 60 frames.

The stem plot on the right depicts the impact of removing individual features on changes in training MSE. The three strongest individual features are the word-time (2), 86th percentile cumulative term frequency, and median term frequency. A similar analysis was conducted by removing different sets of features, and the closed caption features are observed to be the dominant contributor to algorithm performance.

Varying Feature Sets

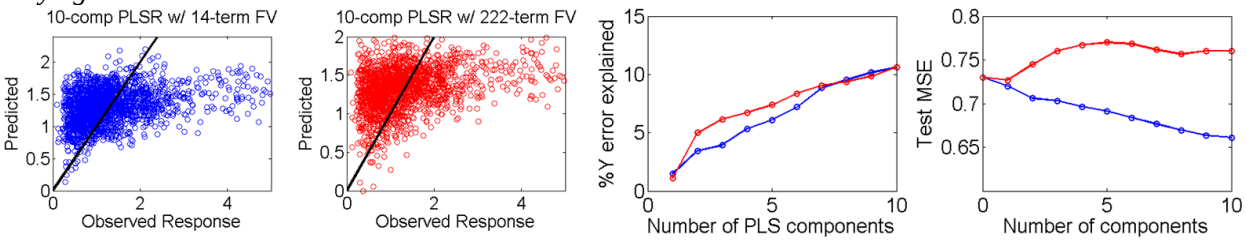


Figure 4: Comparison of 10-component PLS regression using original 14 term feature vector(blue) and expanded 222 term feature vector (red). Predicted vs. observed values seen on left scatterplots. Comparison of relative strength and test MSE shown on the right.

Figure 4 presents a comparison of PLS regression performance on the original 14-term vector and an expanded vector of 222 terms. As seen on the left three graphs, performance is roughly the same for the two algorithms. However, test error is observed to be significantly higher for the larger feature vector, suggesting that inter-feature noise dominated the PLS algorithm's attempt to discern shared information between features.

MSE (Lin Reg)	TF	TF-IDF
Training Err.	.8093	.8063
Testing Err.	.8480	.8353

Table 1: Training and Test MSE using 10-fold cross validation comparing TF and TF-IDF feature performance.

Table 1 depicts a separate analysis comparing the base feature vector to a modified vector replacing term frequency with term frequency-inverse document frequency (TF-IDF). TF-IDF factors in occurrence of words across different videos, and 10-fold cross validation confirms that including this information reduces both test and training error, suggesting that including more features that represent shared information can improve performance.

Conclusion & Future Work

Our results showed that linear regression and SVR find correlations in the data, but are afflicted by the amount of noise in our features and labeling. There was also not enough data for Binary Tree Regression to capture the 2^{14} decision paths generated by our feature set.

Noise issues in data are generally difficult to remove with linear regression algorithms. Reducing noise from our feature set would call for improving the classification of on-screen activity. Relying on OCR simplifies the activity, by failing to distinguish between various cases such as a screen of just text and a screen where the speaker is writing on the board. Improving our labeling would call for better classification for when students are struggling in a lecture. A spike in user interactions might signal issues with the video instead of lecture complexity. Accounting for outliers in both the feature set and labeling would also improve any future algorithms.

Our parameter sweep of frame depth and word depth also provide useful information for future work. The results show that looking beyond the last 60 seconds of video displayed or the last 250 spoken words the performance of our algorithms levels off. Future models could utilize these optimized parameter settings in their implementations.

Regarding parameter selection, the PLSR comparison of the base and expanded feature sets hint that a better feature set could be constructed from reducing noise found in TF/CTF/TFIDF. A moving average TF-IDF feature generator should be evaluated in future studies as a potential vector set, as should more advanced NLP-driven features.

Numerous smaller machine learning projects could be conducted to reduce noise in the final regression model. Particularly, classification algorithms to classify videos by viewership trends would help reduce variance between videos. The OCR extraction also contained a lot of noise, and a separate study could be done to classify the importance of the visuals being displayed in the video.

References

1.) Vapnik, V., Golowich S., Smola A.," Support vector method for function approximation regression estimation and signal processing " , Adv. in NeuralInform. Proces. Syst., 9, 281-287 (1996)
2.) B.Baharudin,L.H.Lee,K.Khan, “A review of machine learning algorithms for text-documents classification”, Journal of Advances in Information Technology1 (1)(2010) 4–20.
3.) Foster, D, et al. “Featurizing Text: Converting Text into Predictors for Regression Analysis” Wharton School of the University of Pennsylvania