2-2-2022

An Automated Zoom Class Session Analysis Tool to Improve Education

Jack Arlo Cannon II
Portland State University

An Automated Zoom Class Session Analysis Tool to Improve Education

by

Jack Arlo Cannon II

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

Thesis Committee:
Christof Teuscher, Chair
John Lipor
Ameeta Agrawal

Portland State University
2022

Abstract

The recent shift towards remote education has presented new challenges for instructors with respect to teaching evaluation. Students in traditional classrooms send signals to instructors which provide feedback for the effectiveness of a given lecture. Virtual learning environments lack some of these communication channels and require new ways of collecting feedback. This work presents a suite of analysis tools for the virtual instructor. Given the transcript and video files for a Zoom meeting, this tool summarizes student sentiment and speaking characteristics. Sentiment scores are derived using state of the art Natural Language Processing (NLP) models. The video file is used to extract interesting features about the lecture content, such as the number of slides, pace of slide changes, and number of words per slide. All metrics were experimentally tested with data from four Zoom meetings, each of which included ground-truth annotations for the slide changes. Transcript-based metrics were validated by comparing to the output produced by Meeting Measures, the project this tool is based on.

A time series outlier detection model was developed for the purpose of identifying slide changes during a presentation. Initially, a percentile-based model performed well on the annotated videos when the optimal threshold was known. However, the process of finding this threshold turned out to be non-trivial for videos without annotations. A Kalman filter model was then tested to alleviate the need for an optimization

step. Ultimately, the percentile-based model was replaced by an HMM-based (Hidden Markov Model) model because of its ability to generalize. When tested on annotated videos, the HMM-based detector performed within a reasonable tolerance of the optimal percentile-based model. Furthermore, for each slide detected an open source Optical Character Recognition (OCR) framework was used to extract text content for computing word counts.

This tool outputs a dashboard containing a set of visualizations for the instructor. These are intended to both identify pain points in the lecture as well as provide a bird's-eye view of general class interaction characteristics. The broader impact goal of this work is to increase remote teaching effectiveness by helping instructors optimize education delivery throughout the academic year.

Table of Contents

List of Tables

List of Figures

1

Introduction

1.1 Motivation

Education delivery has fundamentally changed since the Coronavirus disease (COVID-19) triggered a global pandemic in early 2020. School closures world-wide have forced teachers to conform to a purely remote model of instruction. Though many software programs such as Blackboard and ClassDojo exist to facilitate distance learning, they were ultimately designed for traditional classroom environments [1]. Enter Zoom, a web conferencing program adopted by many schools and universities to provide remote learning environments during the pandemic. Consequently, this shift to remote teaching comes with its own set of challenges for the instructor.

In traditional classrooms, teachers can visually gauge students to determine the general level of engagement. These signals can then be used in real time to make adjustments or ask students directly for feedback. Furthermore, it is usually not possible for students to do other things during a lecture, which keeps distractions low and guarantees some baseline level of attention. Conversely, remote environments provide no such guarantee on attention as students have the ability to do chores, a home workout, or play console games during lecture. In addition, remote teachers are less likely to receive quality, real-time signals from students due to the difficulty of

visually scanning the classroom, especially when many cameras are turned off while many students are participating. Furthermore, these challenges amplify proportionally to the size of the classroom. This lack of feedback makes it particularly burdensome for a teacher to evaluate the effectiveness of a given lecture and is therefore the focus of this work.

1.2 Related Work

1.2.1 Meeting Measures

Washington University researcher Andrew Knight developed Meeting Measures, an R-based tool used to derive metrics from Zoom meetings [2, 3]. The tool produces a number of relevant visualizations, each summarizing a particular metric of interest for the Zoom session. The first version derived transcript-based metrics, which help gauge student interaction by analyzing speech frequency and sentiment. Later versions have expanded to video-based metrics, which take student nonverbal communication into account. At the time of this writing, he has used the tool to administer more than 100 meetings through his website[1] (Fig. 1.1).

When Andrew started Meeting Measures in mid-2020, he had three primary research objectives in mind with respect to virtual meetings [2]:

1. Capture people's behavior in an unobtrusive manner;

2. Provide feedback to attendees on their presence and contributions; and

3. Recommend ways individuals can improve both leadership and engagement.

[1] Currently restricted to those who have a "wustl.edu" email address

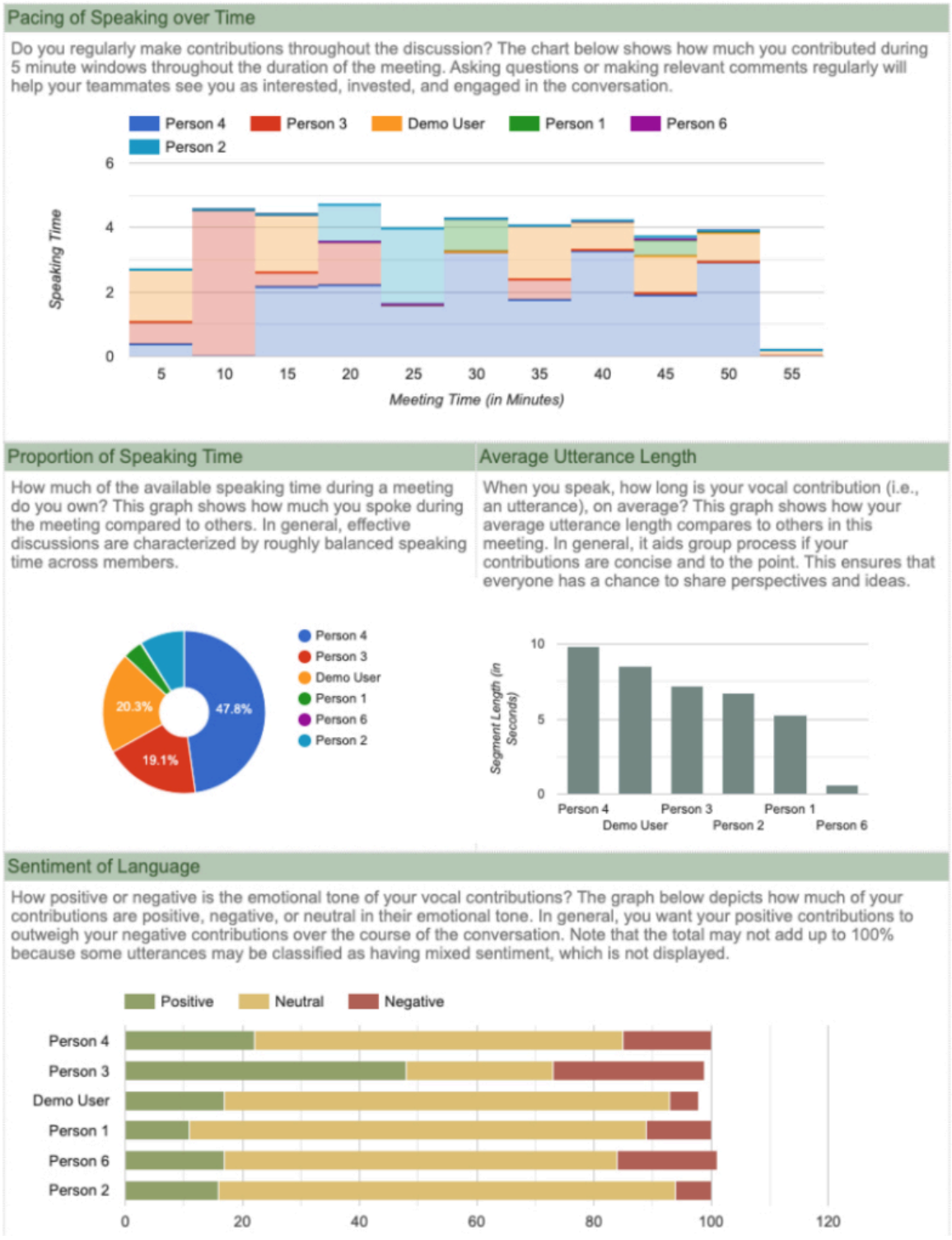


Figure 1.1: A sample dashboard from an early version of Meeting Measures. This set of metrics are derived from the Zoom transcript file. Source: [2]

Individuals who are not affiliated with Washington University can still make use of these tools as they have been open-sourced and made available in an R package called **zoomGroupStats** [3]. The package is well documented, describing in detail how to optimize Zoom settings for data collection and also how to prepare the Zoom files for further processing. Once the data has been adequately prepared, one can follow the guide in the documentation to calculate all available metrics, which result in various datasets that can be used for downstream analysis.

Early versions of zoomGroupStats focused on extracting useful metrics from the Zoom transcripts and chats. Users can build datasets at various levels of granularity. For example, the transcript dataset contains information about each utterance, such as its length (speak time), which is also provided in a summary dataset containing the aggregate speaking time of each individual in the meeting. In addition, there are lexicon-based and machine learning models available for adding sentiment scores to each utterance. The machine learning option requires the user to have a properly configured Amazon Web Services (AWS) account[2]. We used zoomGroupStats to validate and compare our transcript metrics, which is further discussed in 3.1.1.

Recent versions of zoomGroupStats added video analysis capabilities, which also require the user to have an AWS account. Zoom provides a few different recording options for generating video files of the meeting in question. Though it is possible to download multiple video files, the user can only choose one to process. Since video processing is computationally expensive, zoomGroupStats will sample the frames every 60 seconds and save them to disk. The frames are then submitted to AWS Rekognition[3] for face detection and analysis. For each face it detects, Rekognition

[2] `https://aws.amazon.com/`

[3] `https://aws.amazon.com/rekognition/`

returns interesting features such as a gender prediction, whether the eyes are open or not, and an emotion prediction.

1.2.2 Sentiment Analysis

Sentiment Analysis (SA) is a Natural Language Understanding (NLU) task, a subtopic in the field of Natural Language Processing (NLP). At its core, the SA task is a classification problem that maps input text to a sentiment category, such as positive or negative. In this context, the terms "sentiment" and "emotion" are not equivalent. The former is usually binary in nature, while the latter allows for more nuance. For example, the "positive" sentiment category could reflect many different emotional states such as happy, loving, content, or excited.

Traditional sentiment models are built by extracting features from text examples, then training a classifier for the task at hand. Modern sentiment models utilize Transformer-based architectures [4], which yield much better performance. These models work by learning numerical word representations by mapping each word in a vocabulary to a low-dimensional dense vector. The resulting word vectors act as input features to the downstream classifier. One such model, BERT [5], has been foundational to many of the recent advances in NLP, including the models evaluated in this thesis.

Language models such as BERT are designed to learn specific tasks with the goal of capturing the underlying structure of the training corpus. In BERT's case, it was trained for the following two tasks:

- Masked Language Modeling (MLM) and
- Next Sentence Prediction (NSP).

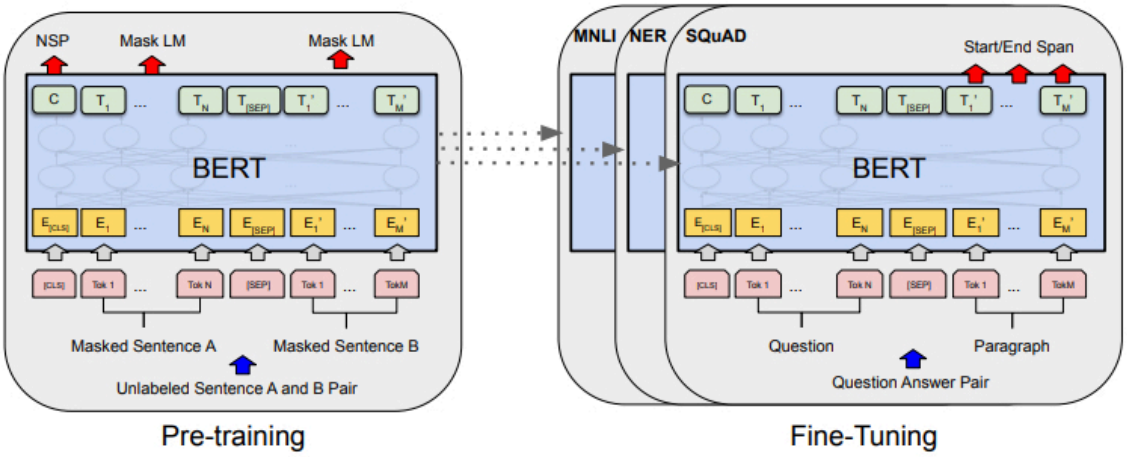


Figure 1.2: The pre-training phase for BERT simultaneously learns the MLM and NSP tasks. The fine-tuning phase adapts the model for specific tasks and corpora. Source: [5]

MLM refers to the task of first presenting a model with a sequence of words (or tokens) where some percentage of them are masked. Then the model learns to predict these masked words. The NSP task presents the model with a sentence, then learns to predict the next sentence in the corpus. BERT was able to learn these two tasks together by structuring the input as a pair of consecutive sentences (see Figure 1.2) [5].

It is important to note that any given language model learns the characteristics of the corpora it is trained on. In the case of BERT, the training corpus consisted of a collection of books (800 million words) [6] as well as a dump of the English Wikipedia (2,500 million words). Therefore, care must be taken when designing a text classification model that utilizes a pre-trained model. For example, models trained on a Twitter corpus will learn much different language characteristics due to the informal nature of Twitter posts. The most common way to address this concern is to fine-tune the pre-trained model's weights using the corpus of interest (Figure 1.2). This is easily accomplished by adding an additional output layer to the model whose weights

will be adjusted for the corpus and/or task at hand. If more accuracy is desired, the weights for additional layers beyond the output layer can be adjusted, which is akin to retraining the model with the new corpus. However this may be unfeasible given hardware and time constraints.

1.2.3 Kalman Filter

The Kalman filter (KF) is an estimation technique developed in the 1960's [7] that is useful for predicting the true state of a process, given noisy measurements. In practice, the KF framework interprets a dataset by assuming each datapoint is a measurement from some process of interest, where the true nature of the process is unknown. It then predicts the next state of the process, along with some uncertainty, and uses the predictions to estimate the next observation. The KF is composed of a dynamic model (Eq. 1.1) for governing the transition between states, and a measurement model (Eq. 1.2) that relates the state of the process to the observation [9, 10]:

$$x_k = Ax_{k-1} + w_{k-1}, \qquad w_{k-1} \sim N(0, Q) \tag{1.1}$$

$$z_k = Hx_k + v_k, \qquad v_k \sim N(0, R) \tag{1.2}$$

Equation 1.1 models the current state of the process x_k, where the w_{k-1} term reflects the assumption that the dynamics of the process includes some Gaussian noise with covariance matrix Q. It should be noted that in most texts, there is an optional *control* term in Equation 1.1 which is not required for this introduction. Equation 1.2 reflects the assumption that the current observed measurement z_k is dependent on x_k, which similarly has a Gaussian noise component v_k with covariance matrix R. The estimation procedure follows a *predictor-corrector* pattern to iteratively improve

predictions [10]. That is, given an initial state x_0 and an uncertainty estimate P_0, the KF will first recursively predict the next set of values then correct those values when the measurement arrives. As a result, the KF is a member of the general class of *Bayesian Filtering* methods [8].

The prediction step is defined by the following equations:

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{1.3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{1.4}$$

Note the superscript "-" in the predictions above, which denote that these predictions represent *priors* for our estimate. Then in our correction step, information is used from the measurement z_k to derive the *posterior* (final) estimates $\hat{x}_k$ and P_k. The equations for the correction step are as follows:

$$K_k = P_k^- H^T (HP_{k-1}H^T + R)^{-1} \tag{1.5}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{1.6}$$

$$P_k = (I - K_k H)P_k^- \tag{1.7}$$

The quantity K above is referred to as the *Kalman gain* and takes on the range $[0, 1]$. It represents the proportion of uncertainty in the previous estimate $\hat{x}_{k-1}$, relative to the uncertainty in the measurement z_k. We can see that K is needed for the two updates (1.6) and (1.7). The quantity $(z_k - H\hat{x}_k^-)$ from (1.6) can be interpreted as the residual between the estimate of the measurement $H\hat{x}_k^-$ and the true measurement z_k. Intuitively, the state update will include more of this residual if K is closer to 1. This follows from the fact that K will be largest when the uncertainty from the previous

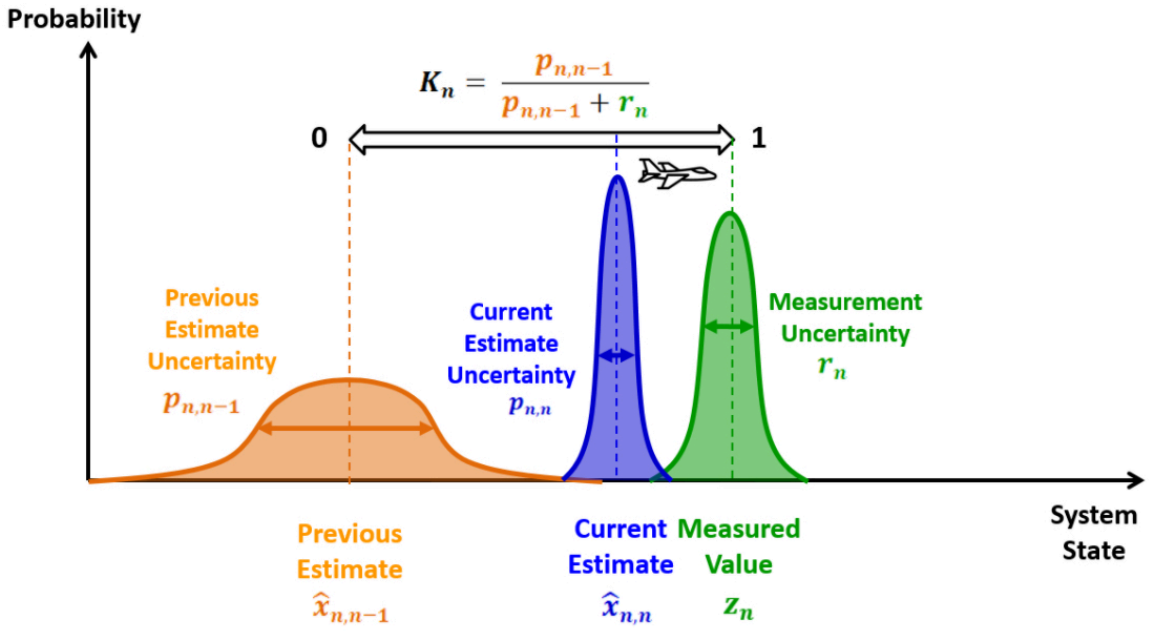


Figure 1.3: Intuition for the high Kalman gain case using an aircraft tracking example. We can see that the previous estimate (orange) has high uncertainty and is therefore far from the actual location of the plane. In this case, the measurement is more accurate (green). Thus a high value of K will cause the current estimate (blue) to move closer to the measurement. Source: [11]

estimate is high relative to the measurement uncertainty. Therefore, the new estimate will be adjusted closer to the measurement (Fig. 1.3). Conversely, a smaller value of K implies more uncertainty in the measurement relative to the uncertainty from the previous estimate (Fig. 1.4). Finally, the update to the uncertainty of the estimate P will decrease at every time step (1.7). The magnitude of K will determine how much P decreases since a smaller K implies less uncertainty in the estimate.

This thesis utilizes the Kalman filter for time series analysis, specifically outlier detection. In that case, each data point in the timeseries represents one of the noisy measurements (z_k). The Kalman filter will estimate the true process ($\hat{x}_k$), which has a smoothing effect on the data (Fig. 1.5). For each estimate, there is some measure of uncertainty (P_k) in the form of a confidence interval band. Therefore, any data point

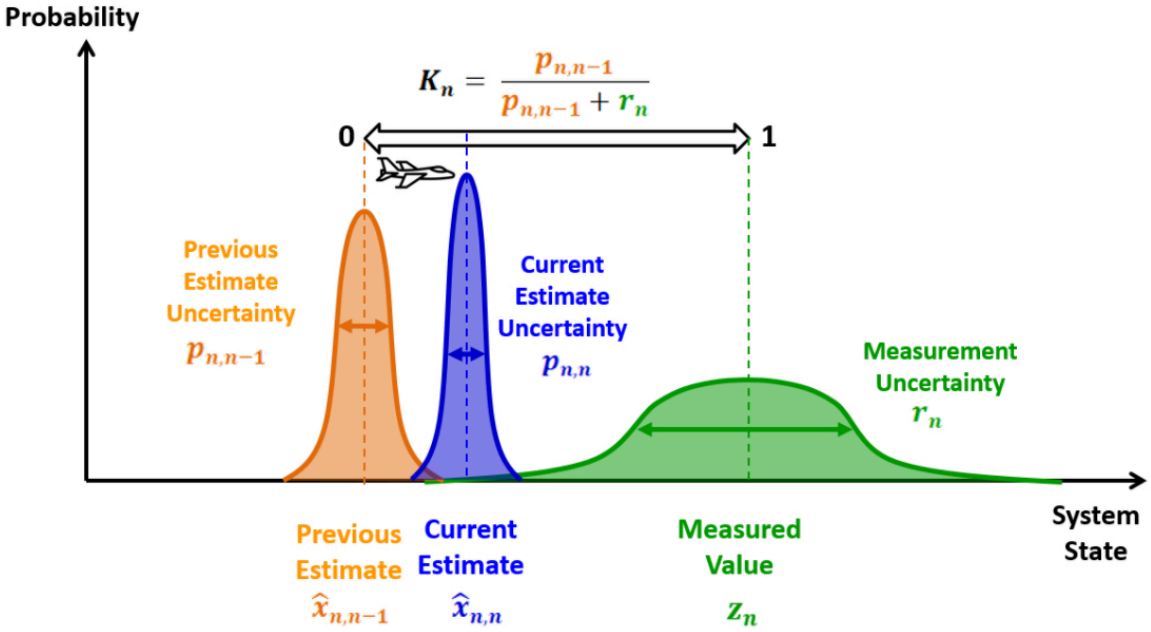


Figure 1.4: Intuition for the low Kalman gain case using an aircraft tracking example. We can see that the previous estimate (orange) has low uncertainty and is close to the actual location of the plane. In this case, the measurement is less accurate with high uncertainty (green). Thus a low value of K will ensure the current estimate (blue) stays close to the previous measurement. Source: [11]

that falls outside of this band could be considered an outlier.

1.2.4 Hidden Markov Models (HMM)

The Hidden Markov Model (HMM) is a generative probabilistic model that produces a sequence of observations, given some other sequence of hidden states which are "hidden" because they are not directly observed. This property makes HMMs useful for modeling the latent space of a particular system or phenomenon. We can describe an HMM with the following components [12]:

- Sequence of T observations $X = x_1 x_2 x_3 \cdots x_T$
- Set of N hidden states $Z = \{z_1, z_2, z_3, \cdots, z_N\}$

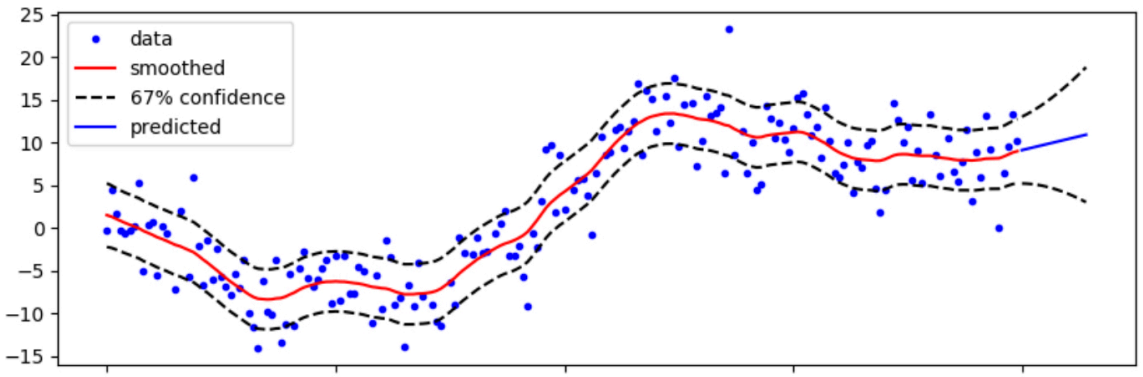


Figure 1.5: A figure taken from the **simdkalman** documentation. A timeseries is smoothed with a Kalman filter, which includes a confidence interval band around the estimates. The blue data points are the noisy measurements (z_k) and the red line is the estimate of the true process ($\hat{x}_k$). Data points outside of the confidence interval bands (P_k) are candidates for outliers. Source: [9]

- Transition probability matrix $A = \{a_{ij} \,|\, 0 < i, j \leq N\}$
- Sequence of T emission probabilities $P(x_t|z_i)$, expressing the probability that observation x_t was generated from state z_i
- A vector of initial probability distributions $\pi = \pi_1, \pi_2, \cdots, \pi_N$, where π_i is the probability that the HMM will start at state z_i

Systems modeled by HMMs assume that at each time step t, an observation x_t is drawn from some probability distribution conditioned on the active hidden state z_i. As the name suggests, HMMs satisfy the Markov property, where the hidden state at each time step is dependent only on the previous hidden state and a_{ij} represents the probability of transitioning from state z_i to z_j. Furthermore, observations depend only on the hidden states that produce them and are therefore independent of the other observations.

Lawrence Rabiner's popular 1989 tutorial defines three fundamental problems for which HMMs are useful for solving [13, 12, 14]:

1. Estimate the optimal sequence of hidden states, given the observations and model parameters (**Decoding**)

2. Calculate the model likelihood, given the observations and model parameters (**Likelihood**)

3. Estimate the model parameters, given only the observations (**Learning**)

For this work, an HMM was used with respect to the first and third problems above. We first fit a model to the observations (learning), which was then used to estimate the optimal state sequence (decoding) for the purpose of flagging outlier observations.

1.2.5 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a computer vision task with the objective of extracting text from images. It has a wide range of applications such as converting pdf documents to machine readable text, traffic sign recognition [15], and developing CAPTCHA-defeating systems [16]. In general, OCR systems work by first preprocessing an image, then detecting blocks of text, and finally by extracting the characters from each block. Advancements in machine learning have led to improvements in OCR engines, as many of them now incorporate deep neural network models. Due to the vast array of images in the real world, OCR performance can vary from image to image. For example, images of license plates or traffic signs will look much different than that of a grocery receipt. As a result, many models achieve performance gains by tailoring to a specific input type.

1.3 Contributions

The primary contributions of this thesis build upon Meeting Measures in the following ways (see section 1.2.1):

- Ported the base transcript functionality from zoomGroupStats to Python and added video analysis tools.
- Implemented plotting tools for visualizing transcript and video metrics.
- Evaluated four open source sentiment models from Hugging Face for the transcript toolset.
- Curated a data set of four Zoom meetings, including ground-truth annotations for slide changes.
- Implemented the following transcript metrics: Average Utterance Length, Proportion of Speaking Time, Speaking Pace Over Time, and Sentiment of Language.
- Implemented a multi-core video processing class for comparing consecutive pairs of frames.
- Evaluated the following outlier detection methods for univariate time series data: percentile, rolling standard deviation, Kalman filter, and HMM.
- Developed an HMM-based model for detecting slide changes in a video presentation.
- Implemented the following video metrics: Slide Count, Pace of Slide Change, and Words per Slide.

- Provided a self-contained tool that utilizes open source models, therefore eliminating dependence on public cloud services.
- Publicly hosted and documented the tool on Github for interested parties to use.

1.4 Document Overview

This chapter motivated the thesis by highlighting the education challenges that have manifested in light of the COVID-19 pandemic. Some background was then provided for the Meeting Measures project, the field of sentiment analysis, the Kalman filter, Hidden Markov models, and Optical Character Recognition technologies. Finally, the contributions of this work were outlined. The rest of this document is laid out in the following manner. The next chapter will describe the implementation details for the tools developed and used in this analysis. Chapter 3 presents the results on a dataset of four Zoom videos. Finally, in Chapter 4 we summarize this thesis and discuss some of the ways this tool can be improved and built upon in the future.

2

Implementation

This chapter focuses on key implementation details during the development of the Zoom analysis tool set. Section 2.1 describes the type of data available for download from Zoom, then defines the metrics that this tool will derive for the user. Sections 2.2-2.3 discusses the datasets that are created from Zoom transcripts as well as video processing details that created the slide change dataset.

2.1 Overview

Zoom allows users to download various components of a given meeting. There are options for saving files locally or to the cloud, with the latter option providing more features. At the time of this writing, users can save the video, chat content, audio, audio transcript, and poll results. Also available for download is meeting metadata and the participant list. In order to derive video metrics, the teacher must also provide the presentation start and stop times for the video file. For the majority of metrics presented in this document, only the audio transcript and video files are required. As a result, the bulk of this section will focus on describing these metrics.

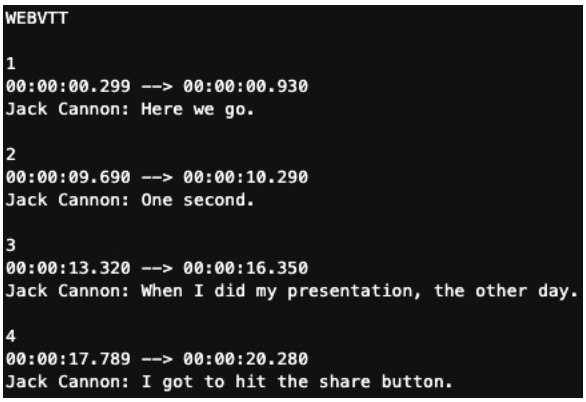


Figure 2.1: The first four utterances of a WebVTT transcript file.

2.1.1 Transcript Metrics

The Zoom transcript is formatted as a WebVTT[1] file where each entry represents an utterance that includes the speaker's name and utterance start/stop times (Figure 2.1). This data is first parsed then used to derive the following set of metrics:

- Proportion of Speaking Time per person (PST)
- Average Utterance Length per person (AUL)
- Speaking Pace over Time (SPT)
- Non-participant List (NPL)[2]
- Sentiment of Language (SL)

SL is different from the other metrics as it requires a sentiment model to quantify a speaker's general sentiment over the course of a meeting. Meeting Measures currently offers both lexicon-based and machine learning sentiment models for this task [3].

[1] `https://w3c.github.io/webvtt/`

[2] Requires the participant list file in addition to the transcript

However, the machine learning option relies on the cloud sentiment scoring service AWS Comprehend[3]. As a result, this thesis derives SL by utilizing the freely available state of the art NLP models at Hugging Face[4].

2.1.2 Video Metrics

The video metrics are intended to provide insight about the lecture content by detecting and inspecting slides in the Zoom video file. These metrics are as follows:

- Slide Count (SC)
- Pace of Slide Change (PSC)
- Words per Slide (WPS)

Slide change detection was accomplished by defining a distance metric (cosine distance) for each pair of consecutive frames. We assume that frames belonging to the same slide will have a small distance, while distances for frames belonging to different slides will be significantly larger. Given this time series of consecutive frame distances, an HMM-based outlier detector predicts the slide changes. On the slide inspection front, an open source OCR framework[5] was used to extract text from each slide to compute WPS.

2.2 Transcript Processing

Transcript files were parsed into a tabular format where each row represented an utterance in a given meeting. Additional metrics were then added to aid downstream

[3] `https://aws.amazon.com/comprehend/`
[4] `https://huggingface.co/`
[5] `https://pypi.org/project/pytesseract/`

meeting_id	id	start	stop	speak_time	speaker	text	num_words	window	window_time	sentiment
83512718053	1	00:00:00.299	00:00:00.930	0.631	Jack Cannon	Here we go.	3	300	0.631	POSITIVE
83512718053	2	00:00:09.690	00:00:10.290	0.600	Jack Cannon	One second.	2	300	0.600	NEGATIVE
83512718053	3	00:00:13.320	00:00:16.350	3.030	Jack Cannon	When I did my presentation, the other day.	8	300	3.030	POSITIVE
83512718053	4	00:00:17.789	00:00:20.280	2.491	Jack Cannon	I got to hit the share button.	7	300	2.491	POSITIVE

Figure 2.2: The first four utterances from 2.1 in parsed form. Additional metrics were then added to complete the dataset for further analysis.

analysis (Figure 2.2). See Appendix B for an example of a complete transcript file.

The resulting transcript dataset contains the following fields:

- **Meeting ID:** The unique meeting identifier containing the utterance.
- **ID:** A unique utterance identifier for a given meeting ID.
- **Start:** The utterance start time.
- **Stop:** The utterance stop time.
- **Speak Time:** The duration of the utterance.
- **Speaker:** The speaker of the utterance.
- **Text:** The text of the utterance.
- **Word Count:** The word count of the utterance.
- **Window:** The time window containing the utterance expressed in seconds.
- **Window Time:** The amount of time the utterance occurred in the window. This will only differ from Speak Time when the utterance spans multiple windows.
- **Sentiment:** The sentiment label assigned to the utterance.

2.2.1 Sentiment Modeling

Sentiment scores were assigned to each utterance in the dataset by using a pre-trained sentiment model. There are hundreds of models to choose from, each of which is trained for a specific language task and domain. For example, FinBERT [17] is a variation of BERT that was fine-tuned on a corpus of financial news articles for the purpose of financial sentiment classification. We selected four popular text classification models from Hugging Face and evaluated them on some sample sentences (Figure 2.3). Two of the models were more consistent than the others, one of these was selected for this analysis. We chose the Typeform[6] model (orange bar) because it had a "neutral" category, whereas the other candidate model (green bar) only supported "positive" and "negative" categories. Since the model outputs three scores for a given input, the sentiment with the highest score was chosen to represent the utterance in question.

2.3 Video Processing

Video metrics require frame-by-frame processing as frame similarity scores are a key component of the slide change detector. It was inefficient to do this sequentially, so we extended the FileVideoStream class from the **imutils** [18] package to perform the task in a parallel manner[7]. In order to reduce the impact of noise from the speaker's camera, a mask was applied to each frame before calculating the similarity (Figure 2.4). The raw similarity scores where then preprocessed into a time series of slide change signals (Figure 2.5).

It is worth noting that video processing is computationally expensive and will result in a long runtime on most laptops. While running this tool on a server with many

[6] `https://huggingface.co/typeform`

[7] `https://stackoverflow.com/questions/61531731/multi-process-video-processing`

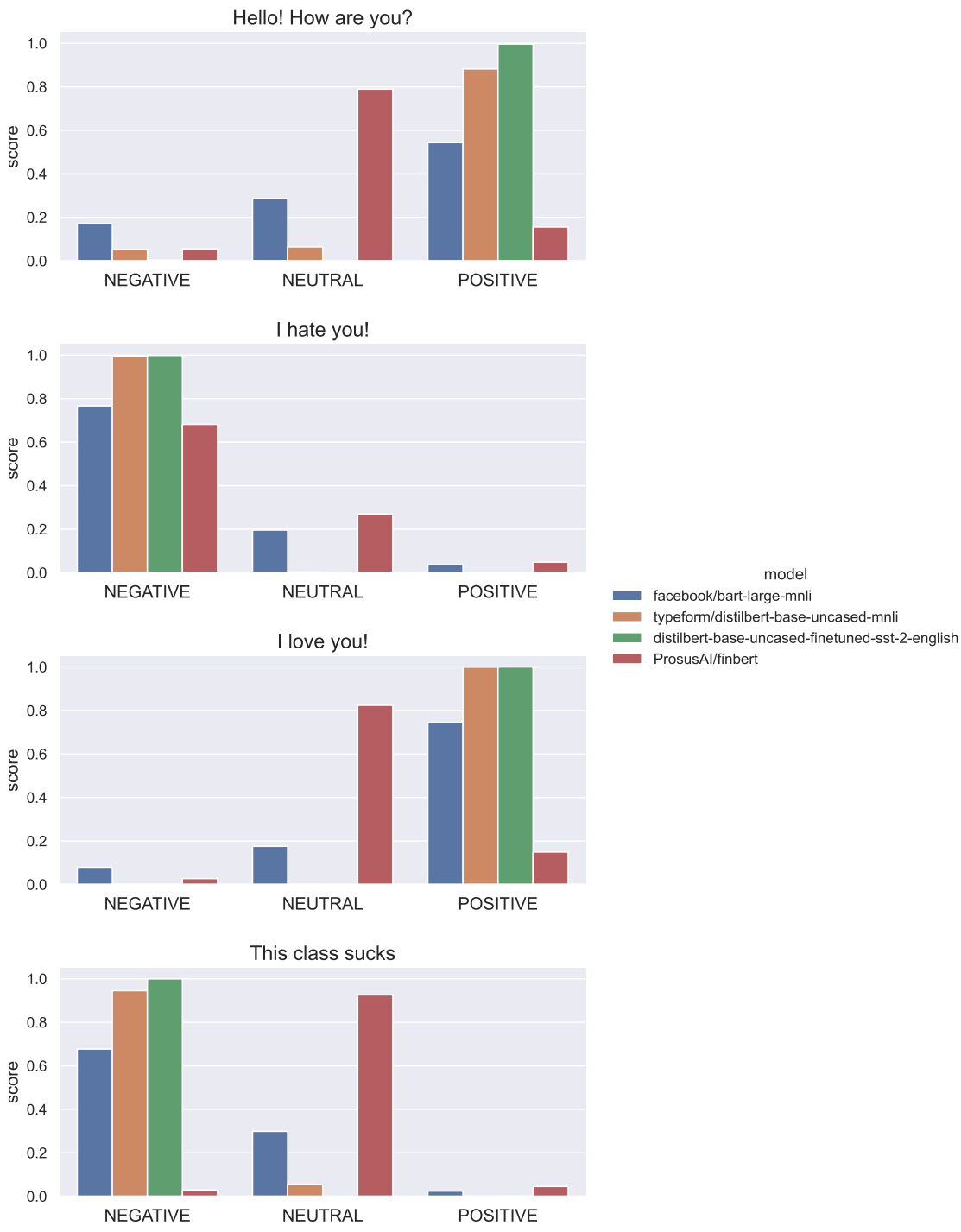
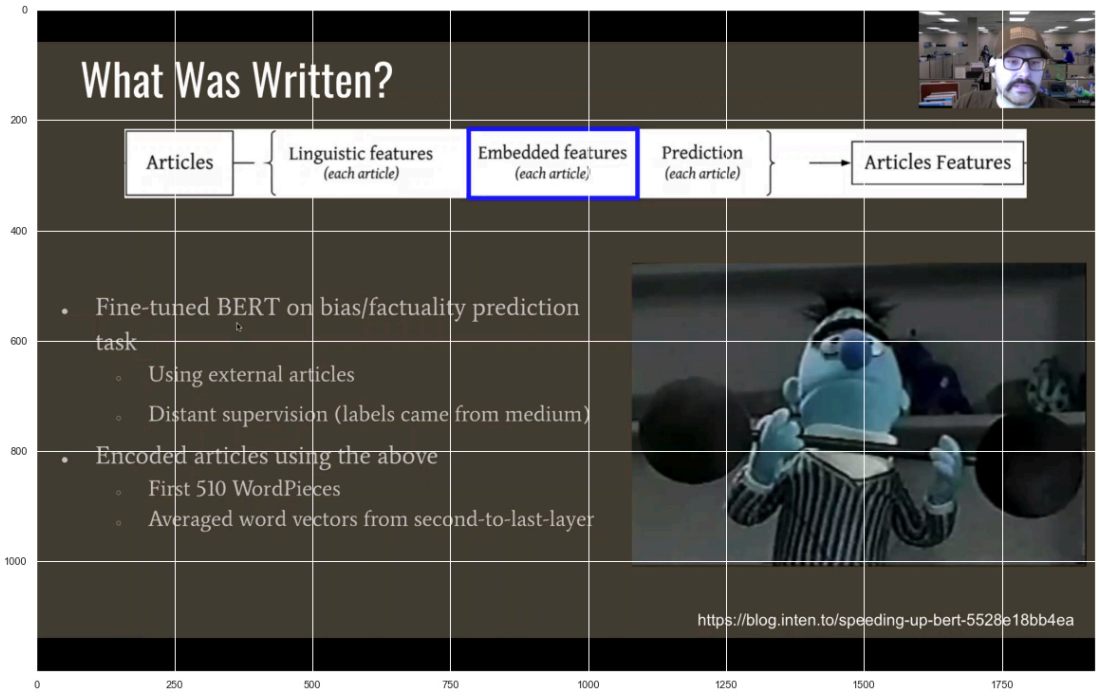
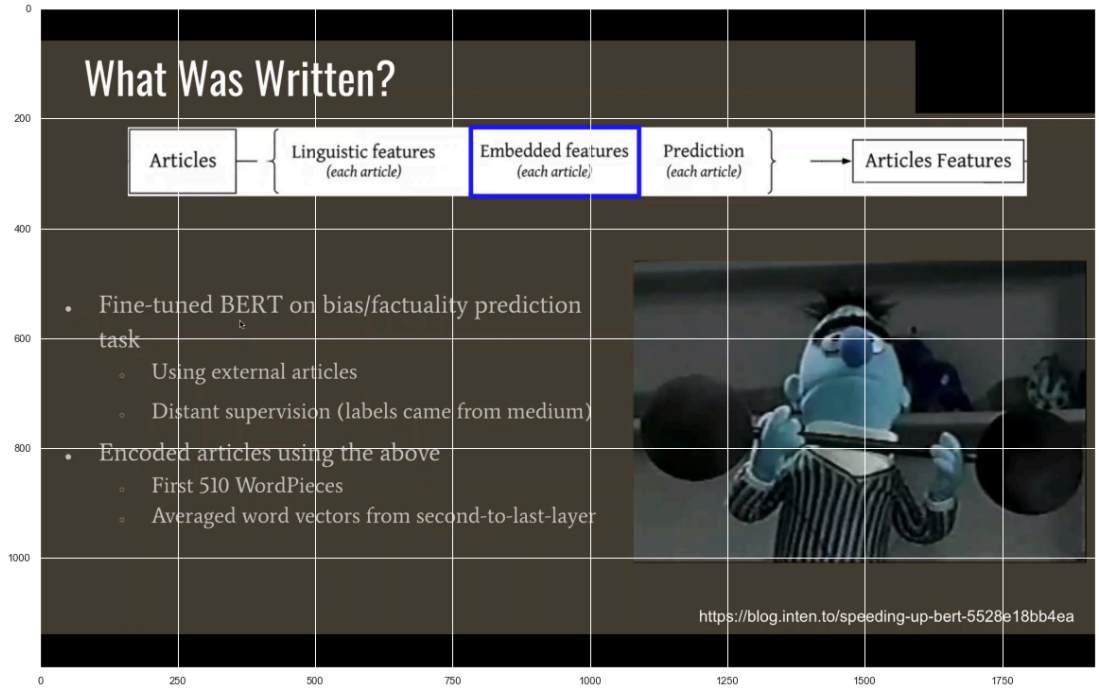


Figure 2.3: Comparison of sentiment models for four example sentences. The scores for each sentiment category can be interpreted as probabilities and will therefore sum to 1. Of the four models, FinBERT (red) was fine-tuned for a specific purpose, which may explain why it was so confused even when applied to extremely polarized test sentences.

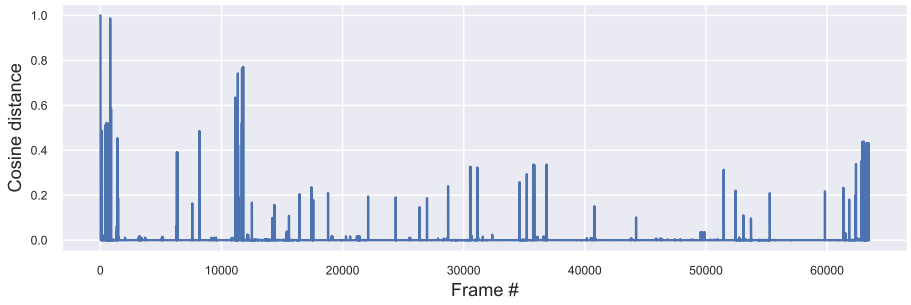


(a) Original Frame

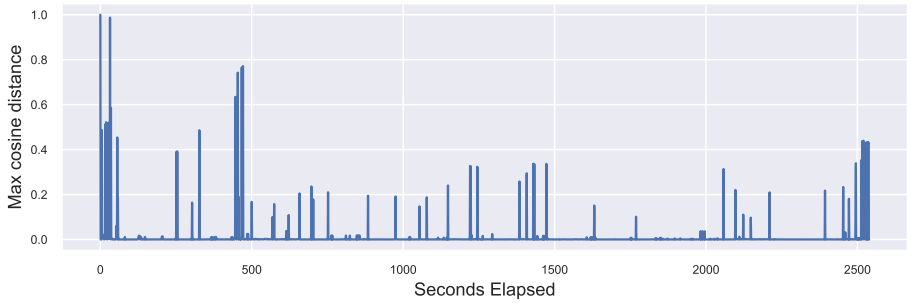


(b) Masked Frame

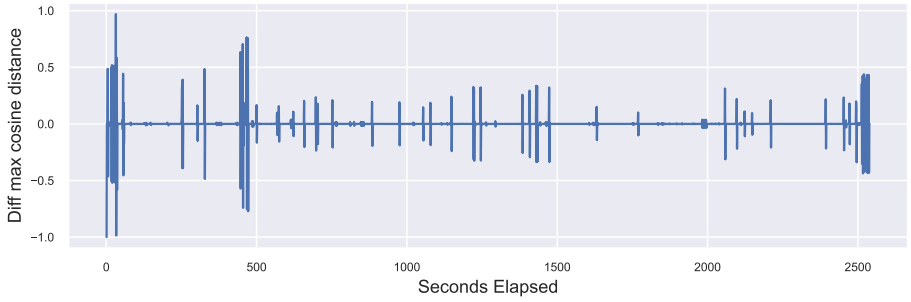
Figure 2.4: Masking the speaker's camera in the top-right corner to remove unwanted noise.



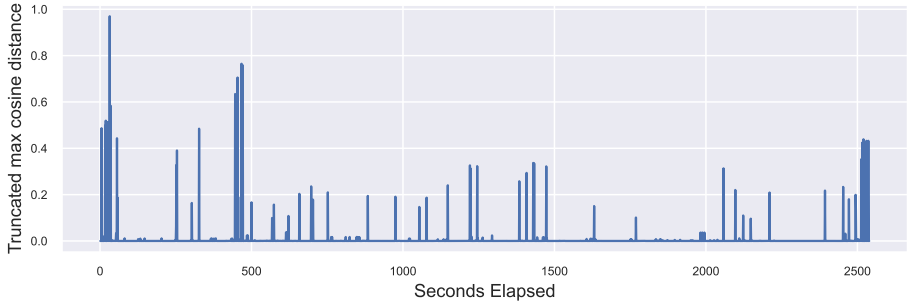
(a)



(b)



(c)



(d)

Figure 2.5: Preprocessing the frame similarity scores. Raw scores for each pair of consecutive frames (a) are first aggregated at the second-level by taking the maximum value for that second (b). The results are differenced to obtain signals for large changes in similarities (c). Each upward spike is followed by an immediate (irrelevant) downward spike. Therefore, the differenced scores are truncated to the range $[0, 1]$ (d).

cores will help, sufficiently long videos will cause a delay on the first run. Subsequent runs will not experience this problem should the user need to regenerate the dashboard for a given meeting.

2.4 Summary

This chapter introduced the idea that Zoom data can be collected for analysis and why it might be beneficial to do so. Transcripts can contain useful information about the speaking patterns of meeting participants. From this, we can derive metrics such as the proportion of time each attendee speaks as well as the length of their average utterance. Videos can be used to analyze slide content for instructors giving lectures. For example, we can calculate the amount of time the presenter spends on each slide, which could help them identify pain points in their lecture style. The next chapter will take a deep dive into the results of applying these tools on some sample data.

3

Results

This chapter presents the results for all the metrics defined in Chapter 2. We begin in Section 3.1 by running one of the datasets through Meeting Measures, then using the results to validate the Python version of the transcript metrics. Section 3.2 describes the process of developing the slide change detector, where sections 3.2.1-3.2.5 discuss the importance of the similarity metric which is then utilized during the evaluation of various models for outlier detection. Finally, in section 3.3 we apply the slide change detector to four sample videos and evaluate the results.

3.1 Transcript Metrics

3.1.1 Comparison to Meeting Measures

One of the initial tasks for this project was to reproduce the Meeting Measures transcripts metrics from Fig. 1.1 - we essentially started to build a Python version of zoomGroupStats. A natural follow-up task was to validate our datasets by comparing them to those produced by zoomGroupStats. We first prepared data from one of our meetings (meeting ID 83512718053) according to the guide in [3]. After running the transcript functions, output datasets were saved and plotted next to plots generated with the Python datasets.

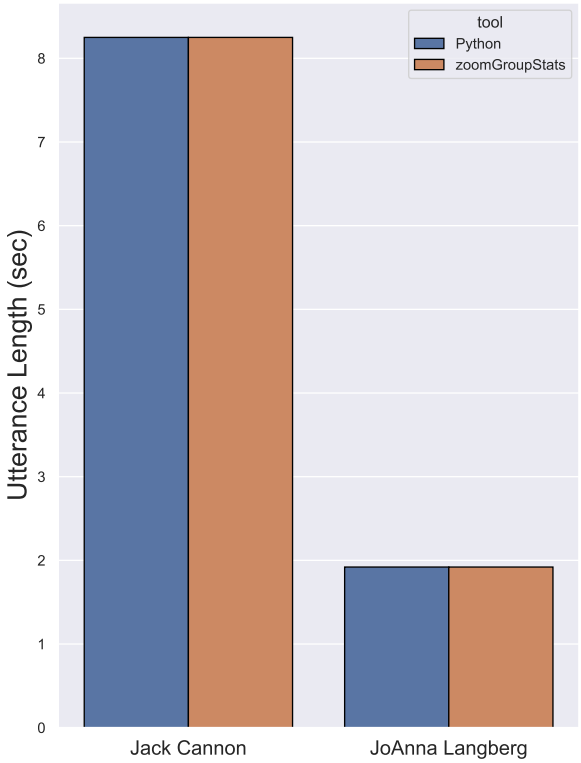
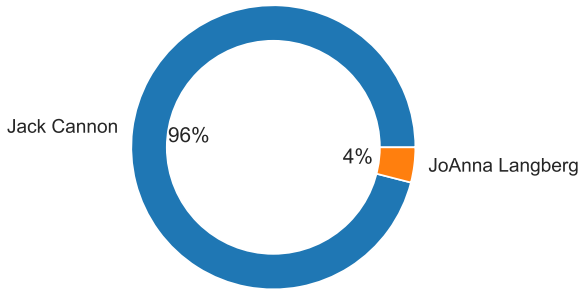
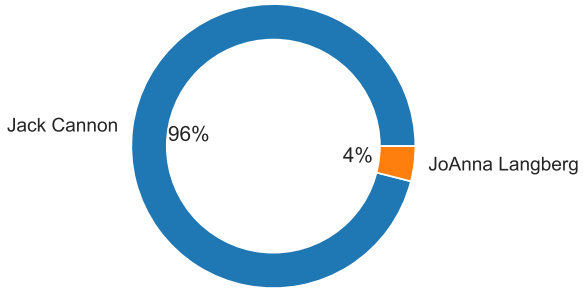


Figure 3.1: Comparison to zoomGroupStats for the AUL metric.

The Python datasets were successfully validated as they were very close to the zoomGroupStats data (some values were identical). For the AUL and PST metrics, there is no noticeable difference between the two datasets. This follows from the fact that both datasets calculated median utterance lengths of 8.25 seconds and 1.92 seconds for the two speakers in the transcript (Fig. 3.1). For speaking proportion, both datasets calculated values of 96% and 4% for the respective speakers (Fig. 3.2). As a result, the plots for these two metrics look identical to the zoomGroupStats version. The SPT comparison showed a very slight deviation for some of the windows with a mean difference of 0.41 seconds (Fig. 3.3), however this did not warrant concern with respect to the validation.



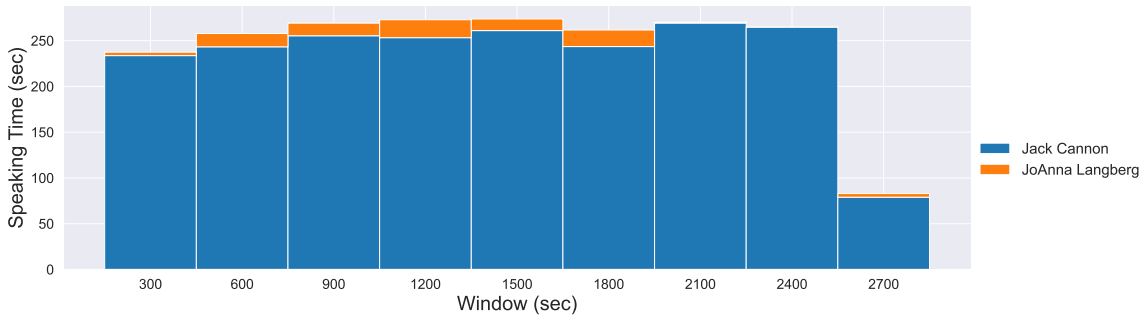
(a) zoomGroupStats



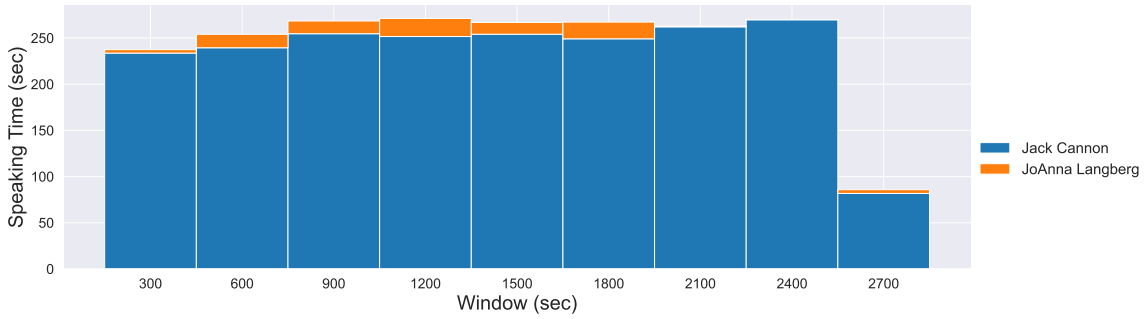
(b) Python

Figure 3.2: Comparison to zoomGroupStats for the PST metric.

Finally, we compared the results for the SL metric (Fig. 3.4), which was the only occurrence of a significant difference in datasets. Here, the discrepancy is wholly due to the choice of sentiment model, with two main observations to note. First, AWS Comprehend returns a "mixed" category, which is not included in the analysis (see Fig. 3.4a). We followed this convention to maintain consistency with Andrew's process, which he describes in the "Sentiment of Language" caption in Fig. 1.1. The second observation to note is how the AWS model tends toward neutrality. The speech in meeting 83512718053 was generally neutral throughout, therefore the AWS model



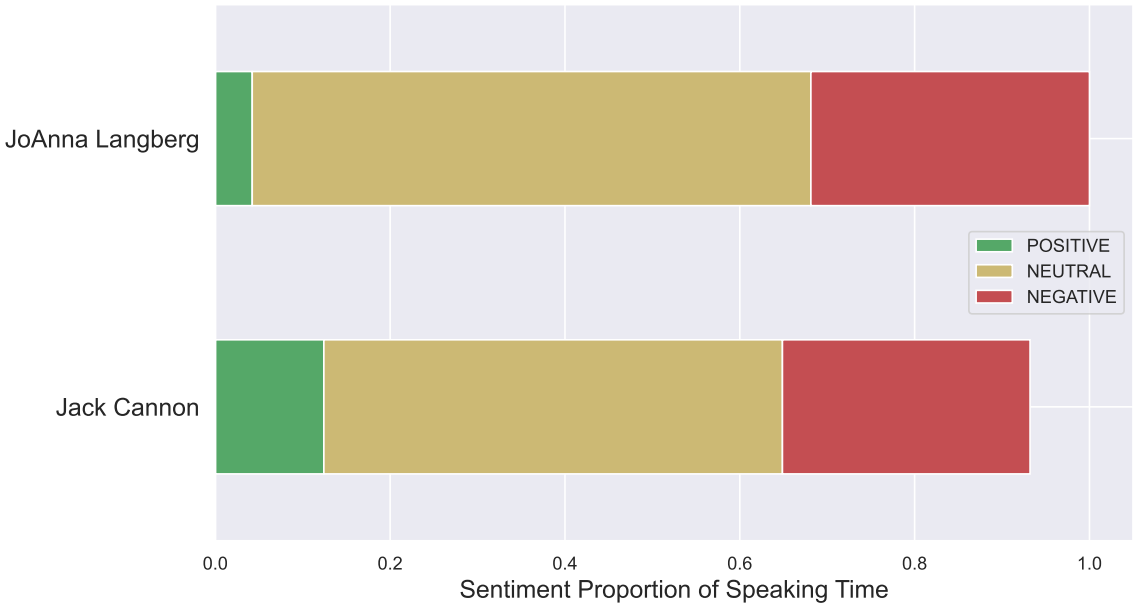
(a) zoomGroupStats



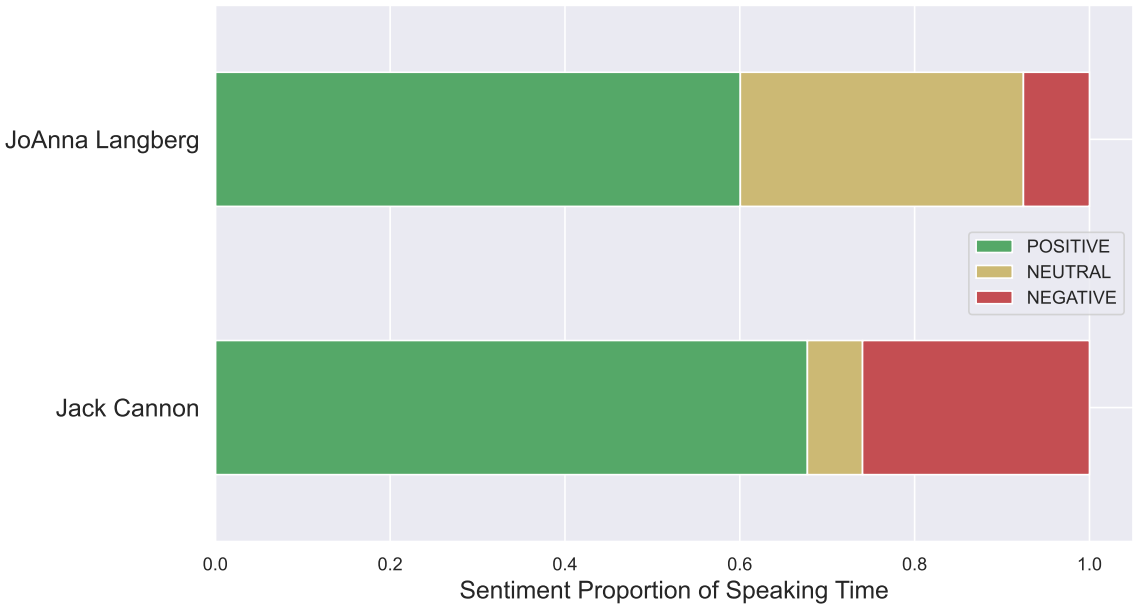
(b) Python

Figure 3.3: Comparison to zoomGroupStats for the SPT metric. Some small differences can be noticed for the 1500 - 2400 windows.

more accurately reflected the true sentiment. This result is not particularly surprising as this model is a commercial service which is maintained by a team of researchers. However, many users may consider the predictions from the Typeform model good enough for their purposes. In addition, many other Hugging Face models can be experimented with should a better result be desired. Furthermore, these discrepancies may not matter to users most concerned with negativity in their meetings, as the two models differ least in that category. Finally, these sentiment models can be more thoroughly evaluated with some example videos where the sentiment is well known, such as when students are extremely negative.



(a) zoomGroupStats



(b) Python

Figure 3.4: Comparison to zoomGroupStats for the SL metric. AWS Comprehend was better able to predict the true sentiment of meeting ID 83512718053 as it was a primarily neutral presentation.

3.2 Slide Change Detector

The slide detection task is an instance of the outlier detection problem. We're looking for characteristics of a video presentation that indicate drastic changes between two frames. However, before that can happen, it is important to define a rule for what constitutes a slide change. Some presentations progress by adding small changes to the slides instead of making each slide completely unique. For example, many speakers will address a number of bullets on a particular slide by starting with a blank slide, then adding the bullets as they talk. In this case, a slide change technically occurs each time a bullet is added to the screen. From a practical perspective, this may not be meaningful enough for those interested in capturing major changes in the presentation material. In this case, we would want to set some minimum threshold on the amount of change between two frames in order to identify relevant slide changes.

Additionally, the teacher may want to exclude sequences of rapid slide changes, such as quickly navigating to a previous slide in order to address a student's question. This kind of action could result in several slide changes in as many seconds, which would be irrelevant for most purposes. In this work, we do not take these nuances into account and instead start with a baseline set of annotations that flag every slide change. Future work may focus on fine-tuning the methods presented in this section to handle situations similar to the aforementioned examples. We experiment with four types of outlier detectors: percentile-based, rolling standard deviation, Kalman filter, and HMM-based. The rest of this section will describe the methods used to develop a model suitable for identifying slide changes in any video.

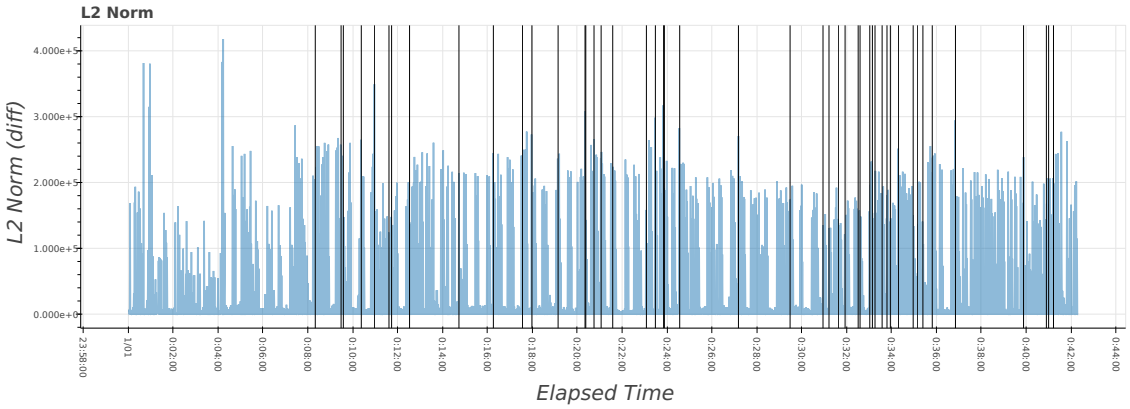


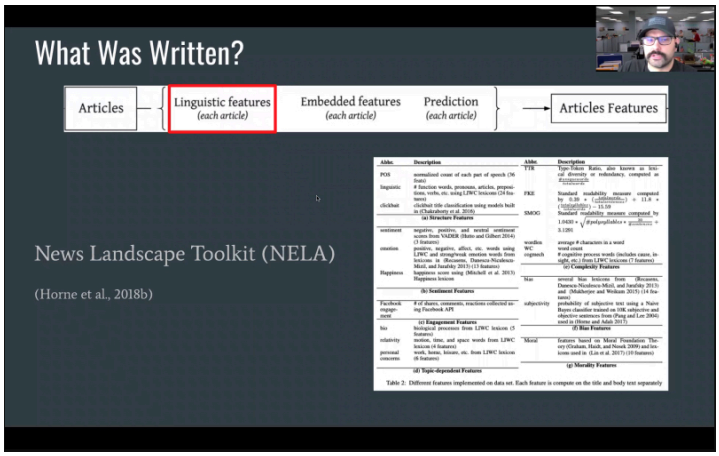
Figure 3.5: l^2-norms diffs for the frames of a sample video act as signals for the slide change detector. The vertical black lines are ground truth slide change annotations.

3.2.1 Similarity Metric

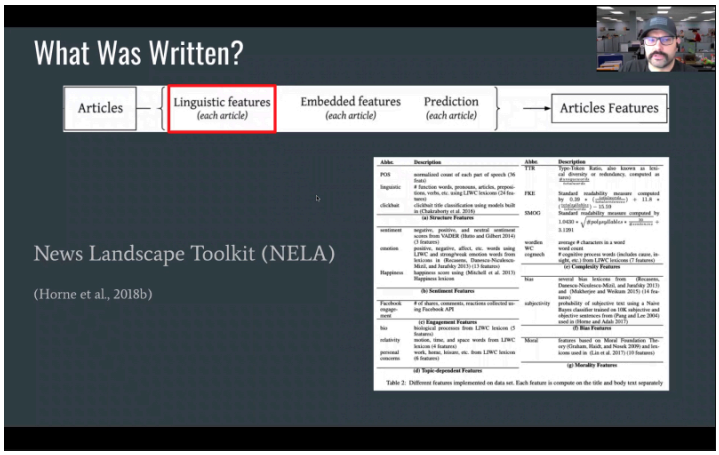
For this problem, we need enough pixels to change between any given pair of frames such that a strong enough signal is generated. One natural measure for this objective is to take a pixel-wise difference then aggregate it into a single metric. We therefore started by calculating a vector of pixel differences followed by taking the l^2-norm to represent our initial similarity measure (Figure 3.5). In this case, it is a dissimilarity measure since a higher value indicates that two frames are dissimilar to each other.

It is clear from 3.5 that there are no concrete slide change signals. Further investigation of the frame differences revealed high levels of distortion in the videos (Figure 3.6, 3.7). This distortion is not detectable by the naked eye, it instead amounts to small pixel translations. Due to the pixel-wise nature of our l^2-norm measure, these translations produced significant amounts of false signals. As a result, we ended up exploring other candidates for similarity measures.

One such metric, *cosine similarity*, works by measuring the cosine of the angle between two vectors in an n-dimensional space and is normalized to the range $[0, 1]$.



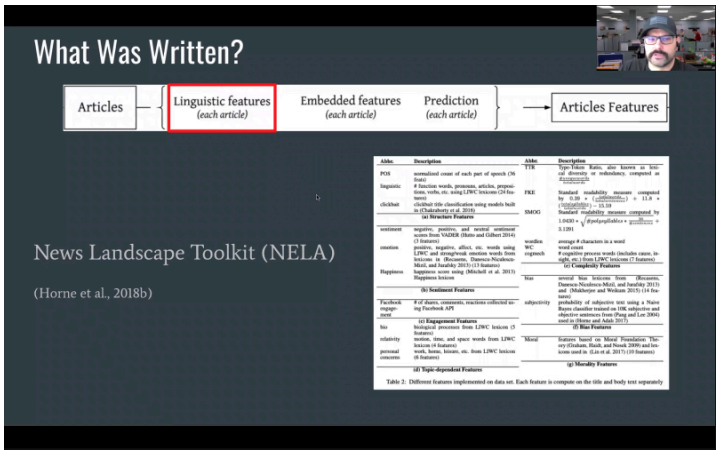
(a) Frame 1



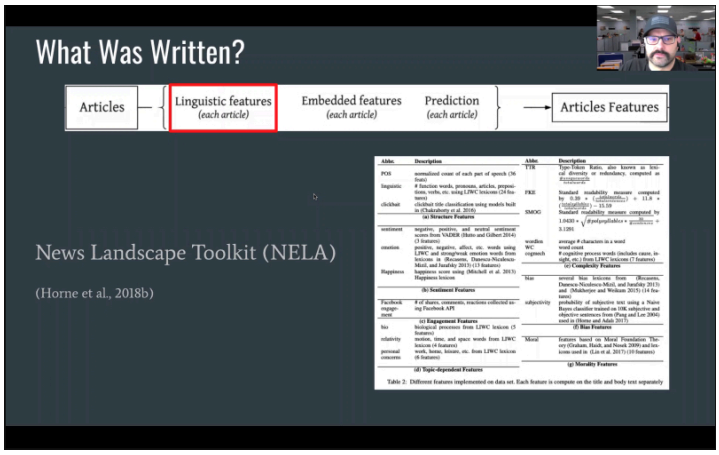
(b) Frame 2

(c) Frame Diff

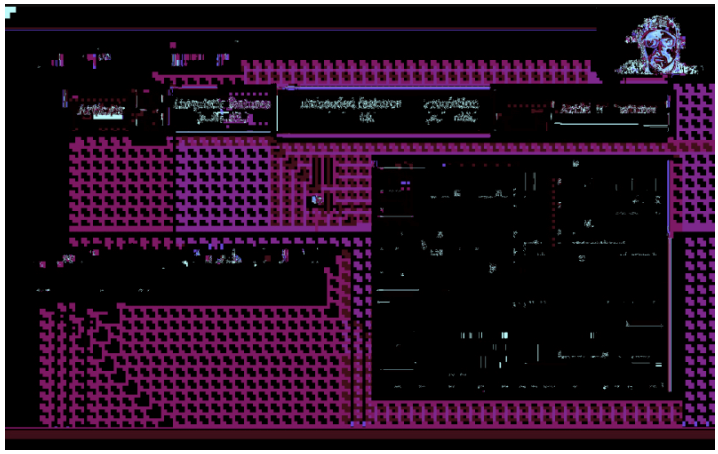
Figure 3.6: Two consecutive frames displaying the same slide and a heatmap representing their pixel-wise differences. For frames without distortion, we would expect to see slight changes resulting from movements in the speaker's camera.



(a) Frame 1



(b) Frame 2



(c) Frame Diff

Figure 3.7: Two consecutive frames displaying the same slide, where the second frame is distorted. Though the frames are visibly identical, their pixel-wise differences are significant and will therefore confuse models based on pixel diffs.

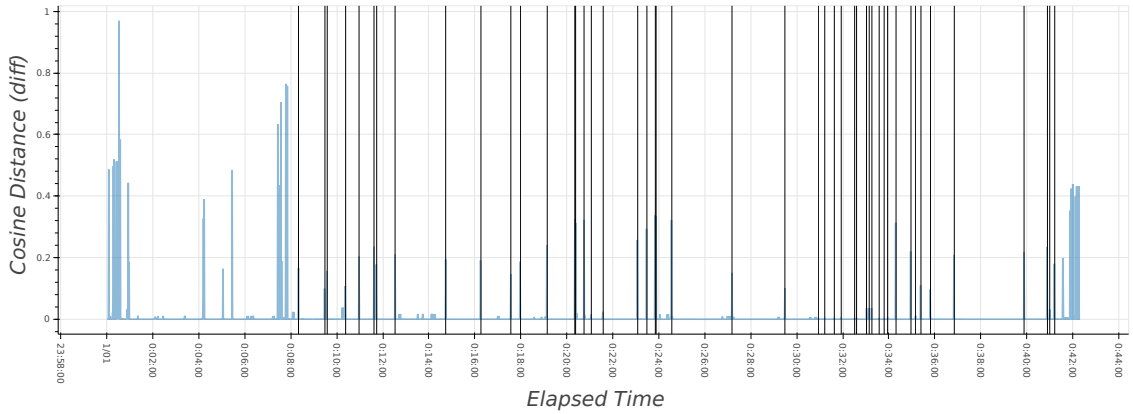


Figure 3.8: The first improvement to our signals from Fig. 3.5 comes by way of monitoring the change in cosine distance between frames. Comparing vector representations of frames proved to be a significantly more powerful measure of similarity.

As a result, vectors with similar values in slightly different dimensions will still be similar with respect to their cosine similarity. In other words, identical images will be recognized as such even when one is slightly distorted. Since we want to measure dissimilarity for our frames, we subtract it from 1 to arrive at the *cosine distance*. Implementing this measure greatly improved our signals, aligning more closely to the ground truth annotations (Figure 3.8).

Though the cosine distance helped greatly, there were still too many false signals in the data. For the Zoom videos used in this work, the speaker's camera is displayed in the upper right-hand corner of the frame. Small movements from the speaker's mouth and head during a presentation may cause extra noise to clutter our signal. Furthermore, in a session where conversations happen often will result in the camera changing many times in a short period of time. This scenario would likely occur often when students ask questions that lead to dialogue with the instructor, which would produce even more noise.

This assumption was confirmed by masking the region of the frame that is generally

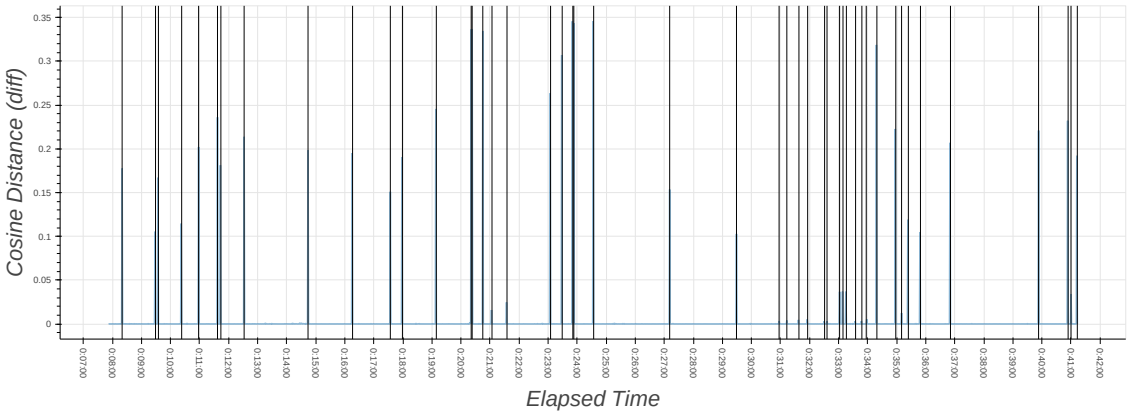


Figure 3.9: A final improvement to the signals from 3.8, which are input to the slide change detector. For some meetings, movement from the speaker's camera will produce false signals. As a result, it is necessary to mask that portion of the frame.

occupied by the speaker. That is, the pixel values were set to 0, thus removing their influence from the cosine distance calculations (Figure 3.9). We found that a fixed proportion of the pixels from the x and y axes selected a region that generalized to all of our test videos, which removed the remaining noise was from the dataset .

3.2.2 Percentile-based Model

A percentile-based outlier detector makes predictions based on some prespecified percentile threshold. For any timeseries of interest, this threshold is found by choosing a percentile q, then computing the value corresponding to the q-th percentile[1]. Next, any datapoint in the timeseries greater than or equal to this q-th percentile value is flagged as an outlier. For example, a percentile of $q = 50$ would flag any datapoint greater than or equal to the median, since the median of a dataset is the 50-th percentile. This of course leads to the problem of finding the percentile threshold that minimizes the prediction error. Unsurprisingly, each of our test videos required

[1] `https://numpy.org/doc/stable/reference/generated/numpy.percentile.html`

different thresholds for optimal performance (Figure 3.11). We use the F1 score for evaluation since it takes both precision and recall into account, which is desirable for unbalanced classification problems such as this one (Figure 3.10).

There are two interesting observations from Figure 3.11. First, the optimal thresholds are extremely sensitive such that small changes in either direction will result in a sizable decrease to the F1 score. Figure 3.12 demonstrates the differences in accuracy resulting from a $\pm$ 1% change in the threshold. This property makes any attempt to generalize threshold selection especially difficult (e.g., using the mean). Second, some videos require the frames to be masked in order to perform well. For example, the video with meeting ID 170127 consists of a few long dialogues with students while answering questions. As a result, the camera transitioned enough to render the model worthless.

These experiments show that percentile-based models perform well when an optimal threshold is known before hand. However, the choice of threshold value does not allow room for error. Since the task of finding an optimal threshold is non-trivial for a new video without ground truth annotations, percentile-based models are infeasible for this use case. Therefore, another technique is needed, which we'll benchmark against these optimal percentile-based models.

3.2.3 Rolling Standard Deviation Model

The percentile-based model is a good baseline from which to compare results from further experiments. Since that model relies on global statistics to make predictions, it seems natural to investigate methods that make predictions based on local statistics. One way to accomplish this is to track the rolling mean and standard deviation over some number of periods in the past. Then when the difference in cosine distance

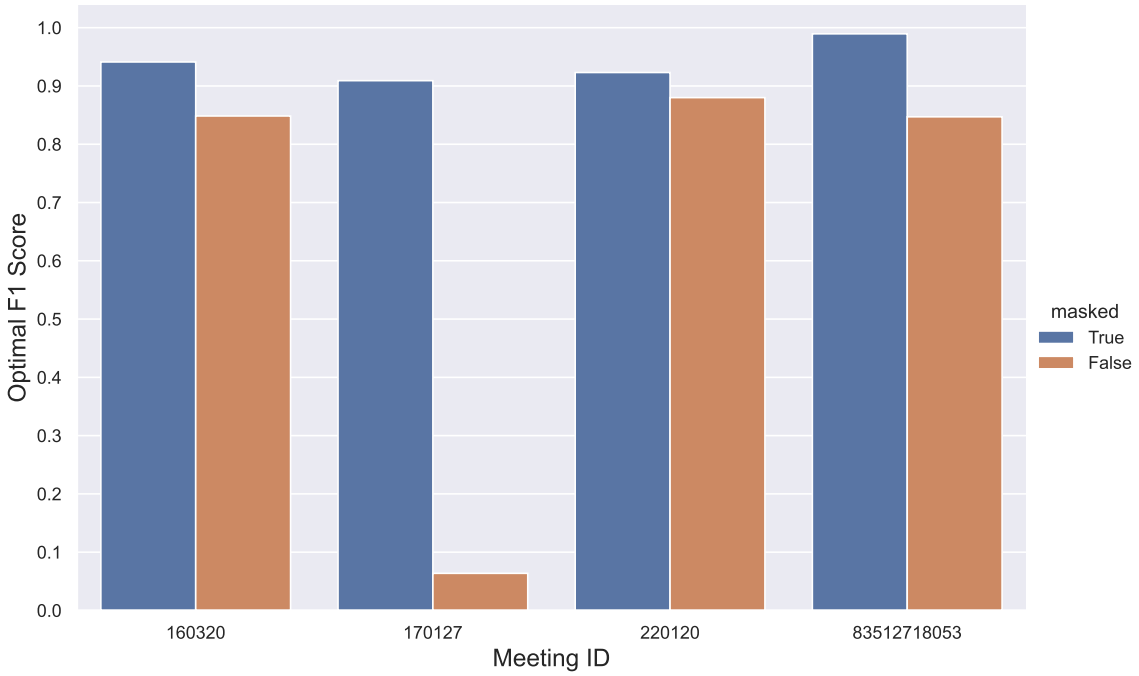


Figure 3.10: F1 scores for the four test videos, each evaluated at their optimal threshold for both masked (blue) and unmasked cases (orange). During the presentation for meeting 170127, students asked questions on more than one occasion, which resulted in frequent changes to the camera portion of the frame while holding the slide constant. Furthermore, this particular presentation had incidents where the instructor rapidly cycled through slides to revisit prior slides.

deviates from the mean by a certain magnitude, we predict a slide change at that time step.

There are two parameters of interest for this model: window length (w) and number of standard deviations from the mean (s). Immediately we see that one choice of parameters will have a different impact on predictions for different videos (Fig. 3.13). Similar to the percentile model, the optimal choice of parameters is not known beforehand and must therefore be optimized. For example, if we increase the window length to 20, the number of standard deviations from the mean must almost double in order to get visually similar results (Fig. 3.14). This fact was discovered by trial and error, an approach that does not scale well. In addition, the predictions are extremely

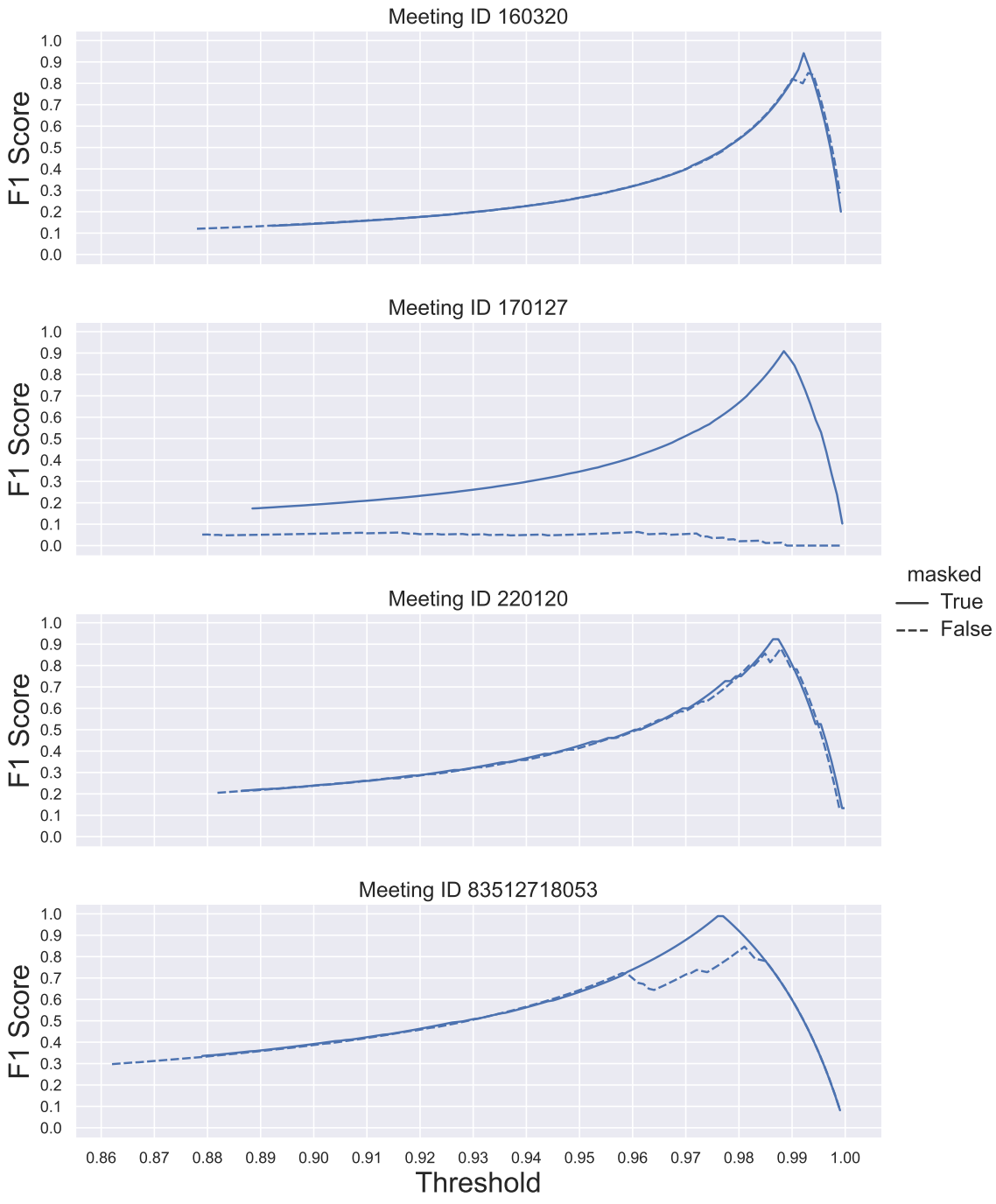
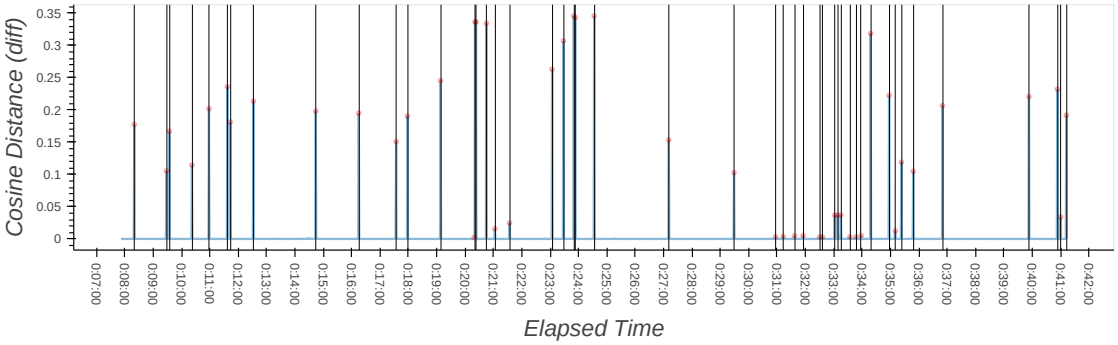
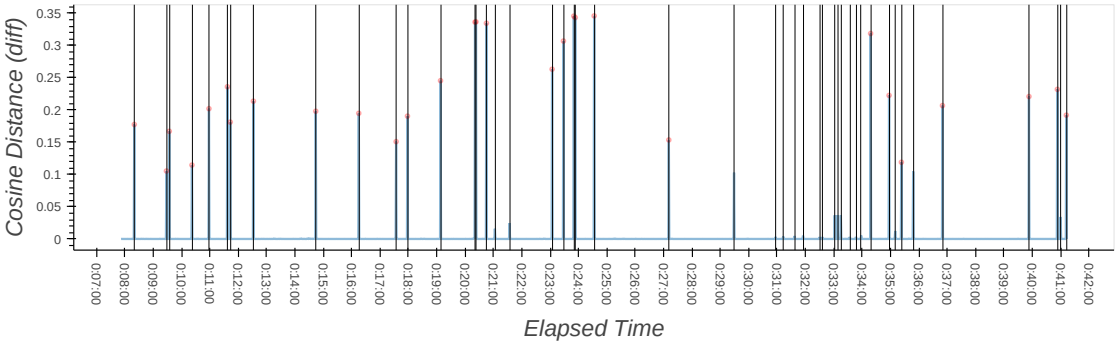


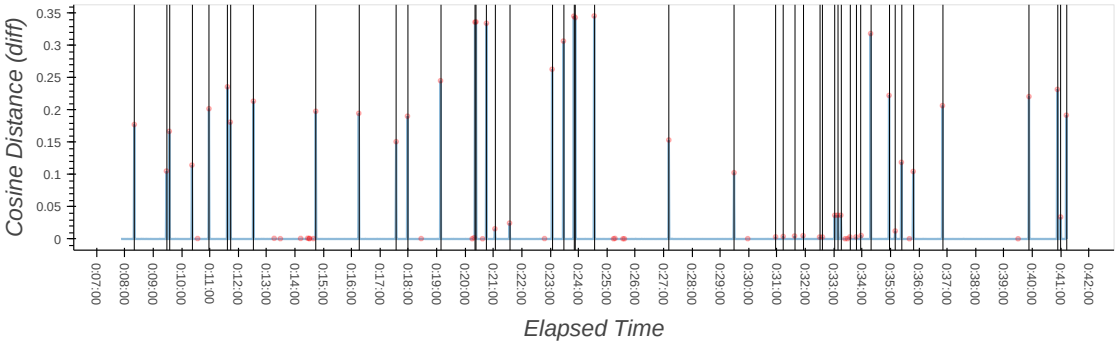
Figure 3.11: F1 scores for each of the four test videos, by percentile threshold. The models perform well when the optimal threshold is known and the camera is masked (solid line). However, these models are sensitive to changes in the threshold such that a small move in either direction would result in a significant performance decrease.



(a) Optimal threshold

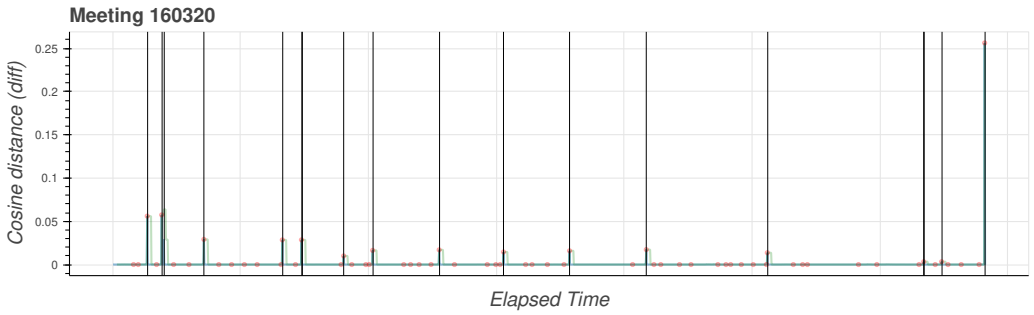


(b) High threshold (+1%)

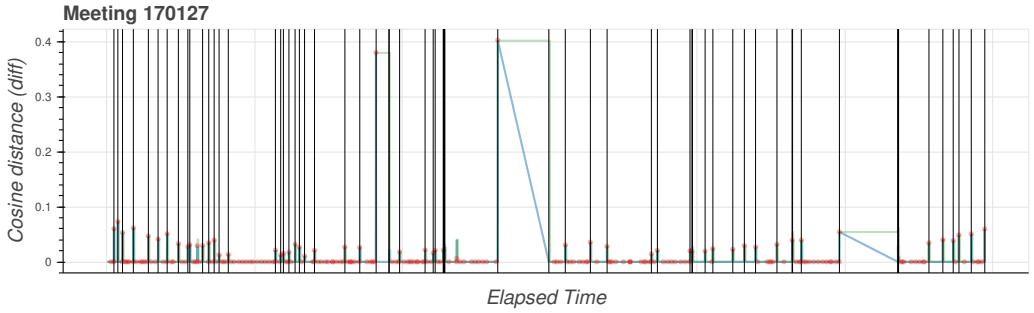


(c) Low threshold (-1%)

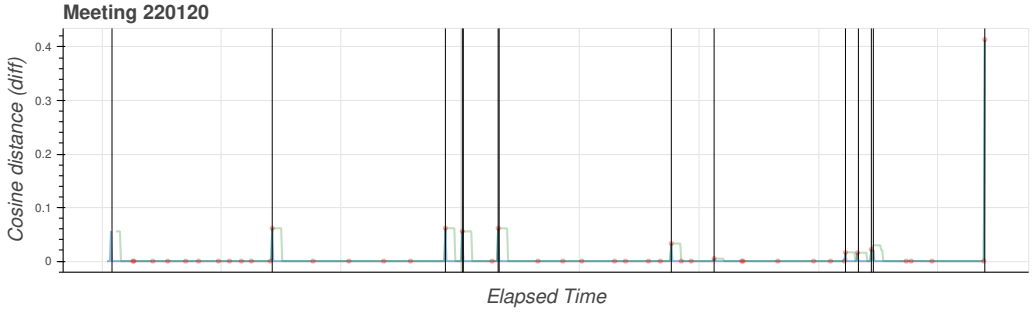
Figure 3.12: Percentile-based model predictions for the optimal, high, and low cases (Meeting ID 83512718053). Ground truth annotations are represented by the black vertical lines and predictions represented by red circles. A higher threshold results in more false negatives while a lower threshold results in more false positives. Here we see the impact resulting from a small deviation in percentile.



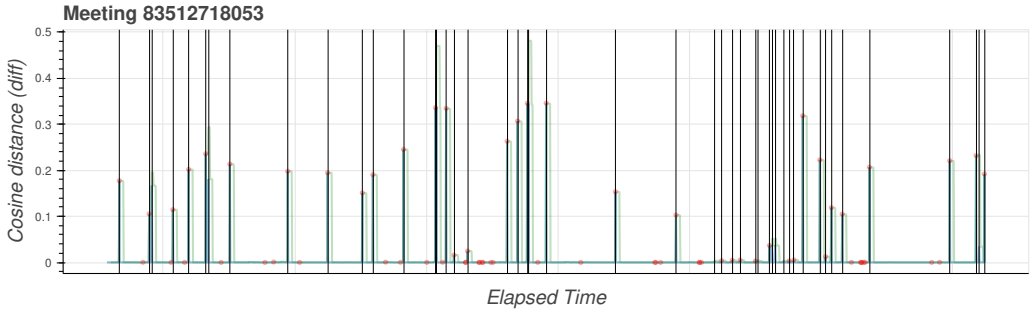
(a)



(b)



(c)



(d)

Figure 3.13: Rolling standard deviation model predictions for all four videos ($w = 10$ and $s = 2.84$). The standard deviation band is denoted by the green line, which is noticeable only during a spike in the difference in cosine distance. The variation in the data is tight enough between spikes to cause false positives. Meeting 170127 (b) has significant amounts of false positives.

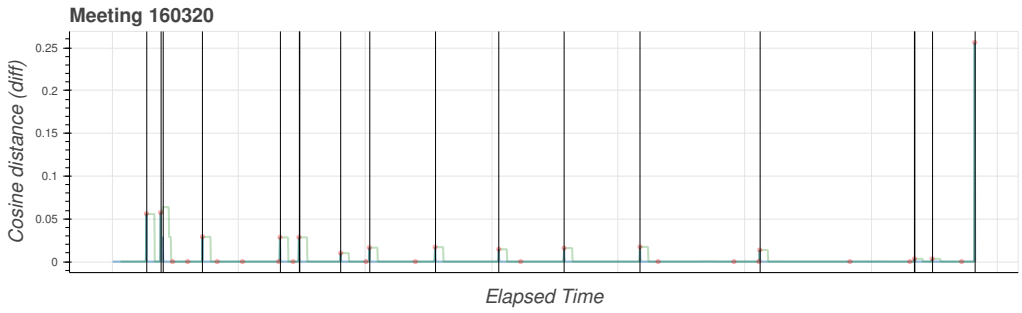
sensitive to parameter changes. It is standard for s to be require precision to the one-thousandth decimal place and beyond, much like the percentile model (Fig. 3.11).

This model suffers the same drawbacks as the percentile model and essentially behaves like a "rolling percentile" model, where local statistics cause the same problems as the global statistics. First, the parameters must be optimized before the model can make useful predictions for a given time series, which is not desirable due to the lack of generalization. Second, predictions based on standard deviation thresholds do not perform well when the data has extremely low variation (Fig. 3.15). As a result, any threshold-based method needs to produce a band that stays low enough to trigger a prediction without being influenced by areas of low variation. Therefore, we move on to experiment with filtering/smoothing methods with this objective in mind.

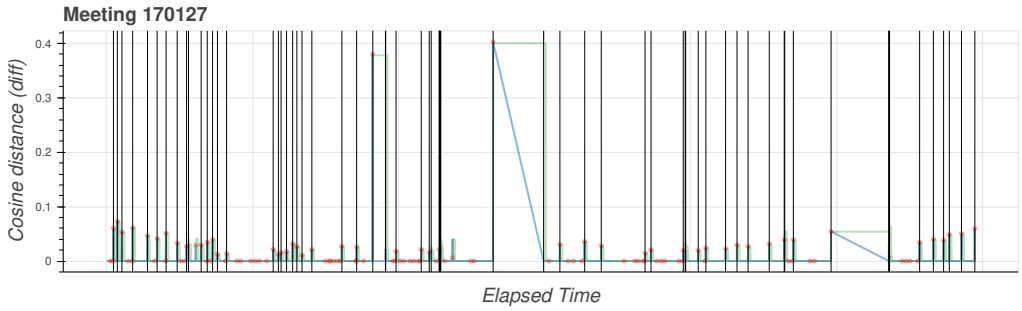
3.2.4 Kalman Filter Model

The Kalman filter is a common tool for time series outlier detection and comes with the added benefit of dynamic prediction. That is, it can adjust its prediction with each new data point it receives. For this experiment, we made use of the **tsmoothie**[2] Python package. We started by exploring the impact from two parameters: the noise level of the process (pn) and the number of standard deviations used to calculate the band ($nsig$). We can see that the initial parameter values ($pn = 0.5$, $nsig = 2$) produce a band that hovers at a more favorable distance for our purposes (Fig. 3.16). A nice feature of this Kalman band is the spiking that occurs with spikes in the underlying, which will ensure that a signal must be sufficiently large to warrant a prediction. However, the hovering distance clearly varies with the meeting. While the prediction threshold looks reasonable, it is too high for some of the signals in meetings

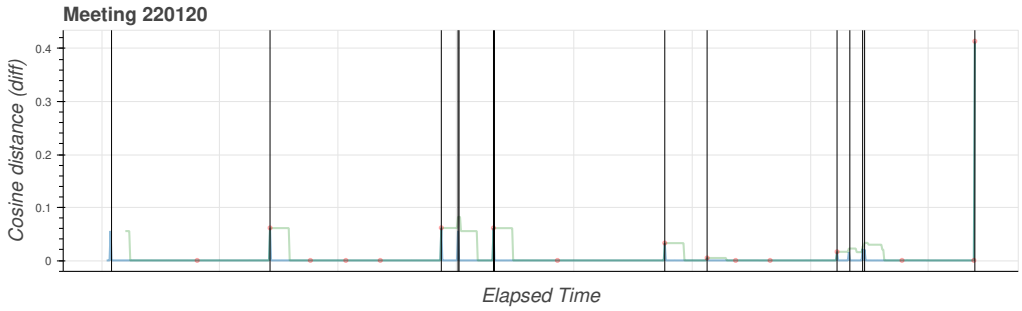
[2] `https://github.com/cerlymarco/tsmoothie`



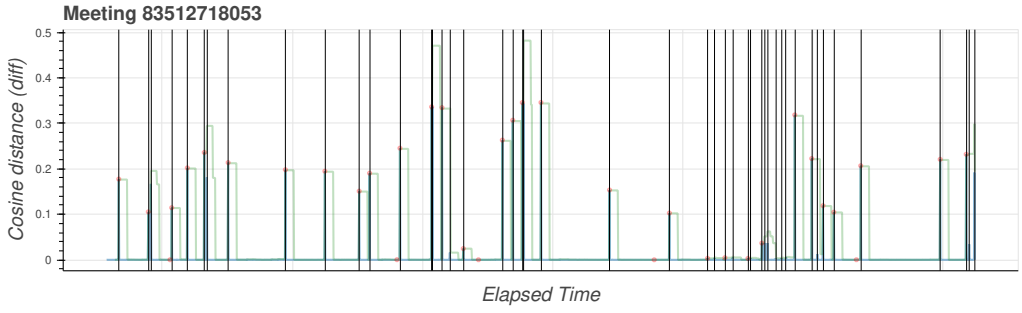
(a)



(b)



(c)



(d)

Figure 3.14: Rolling standard deviation model predictions for all four videos ($w = 20$ and $s = 4.22$). The standard deviation band is denoted by the green line, which is noticeable only during a spike in the difference in cosine distance. The variation in the data is tight enough between spikes to cause false positives - this is most noticeable for meeting 220120 (c) where there are 3 mispredictions in the second quarter of the dataset.

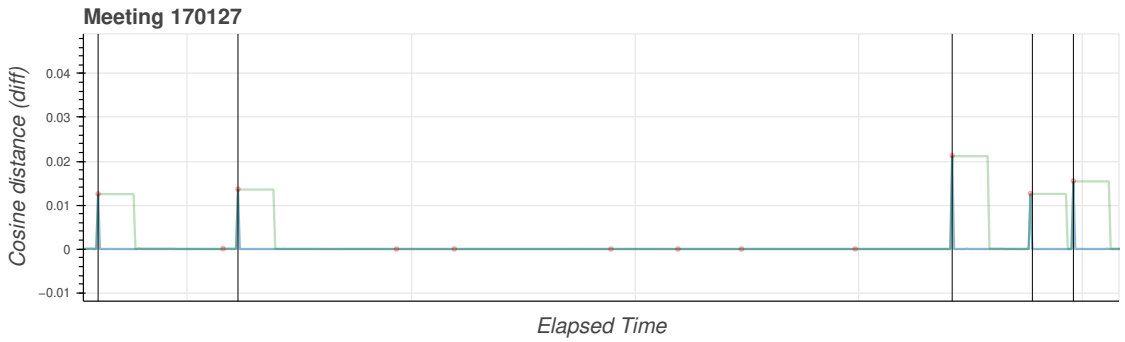
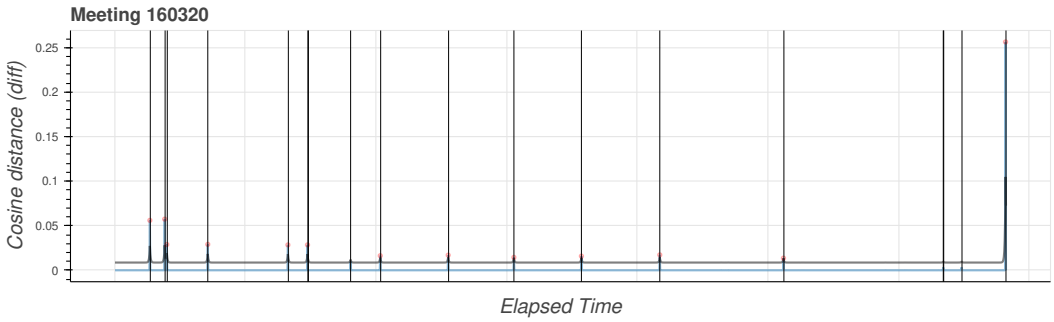


Figure 3.15: An enlarged section of the meeting 170127 time series from Fig. 3.14 ($w = 20$, $s = 4.22$). Low variance data will cause mispredictions even with very high values of s. We can see there are six false positives in the center portion of the figure. The green band is not noticeable at > 4 standard deviations from the mean, which indicates the variance of the data during this interval is too low for a rolling standard deviation rule to be feasible.

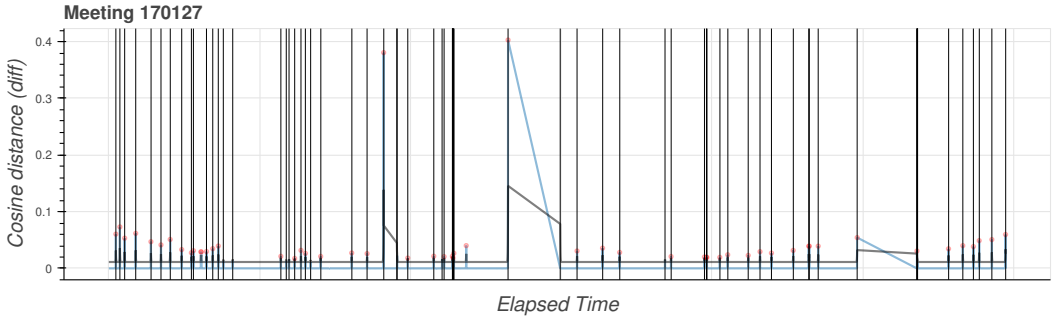
220120 and 83512718053.

Since we know that these time series are generally of low variation, we reduce $nsig$ to 1 and inspect the results (Fig. 3.17). We hold pn constant because this data is not particularly noisy, therefore the primary driver of the predictions is the distance from the threshold to the data. There is a noticeable decrease in the hovering distance of the Kalman band, which was consistently favorable for each meeting. These results are encouraging, which leads one to be curious on how low the band can get before mispredictions start to occur. For this dataset, that threshold corresponds to an $nsig$ value of 0.03.

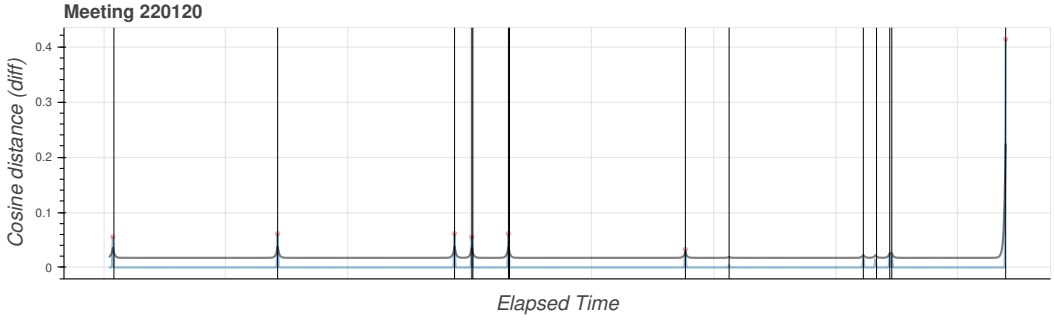
Given prior domain knowledge of Zoom frame similarities, we can make an informed decision on how to set pn and $nsig$. As previously mentioned, this data is not noisy relative to the number of data points. In fact, we can look at this problem as one of trying to detect the noise when it does occur. As a result, the standard value of pn will work for our purposes. In order to set $nsig$, we want to pick up as many signals as possible without producing false positives. We know the variation of the data is



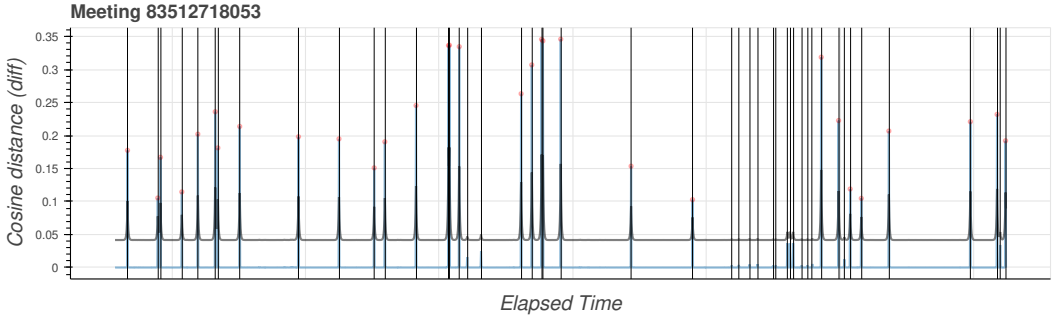
(a)



(b)

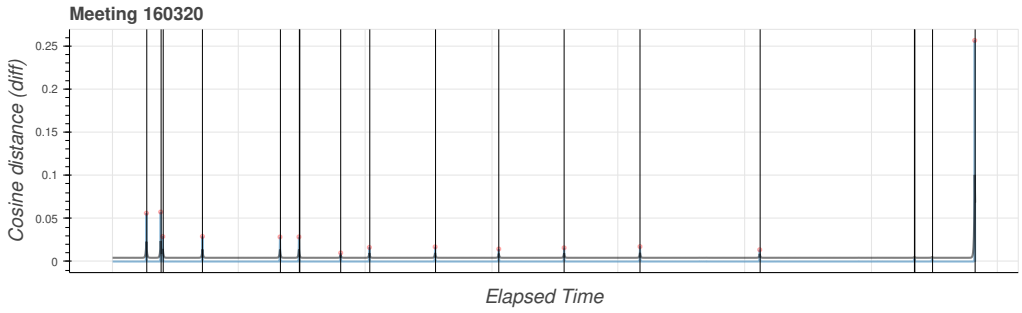


(c)

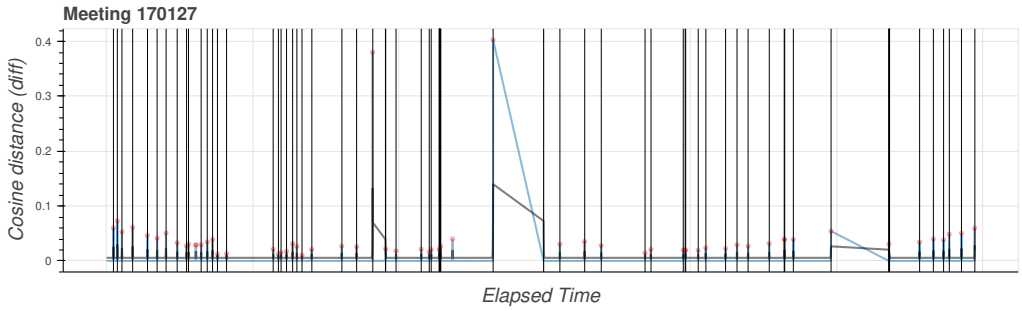


(d)

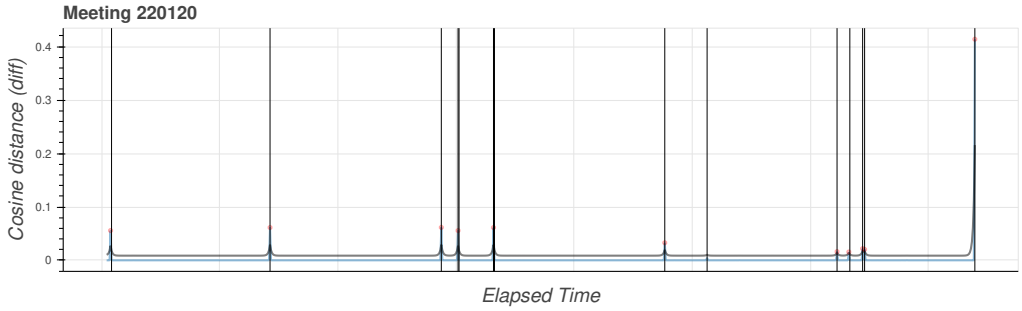
Figure 3.16: Initial Kalman filter predictions for the four meetings ($pn = 0.5$, $nsig = 2$). The prediction threshold is denoted by the black line tracking the cosine distance diffs. Unlike the rolling standard deviation case, the Kalman filter produces a band that hovers away from the data.



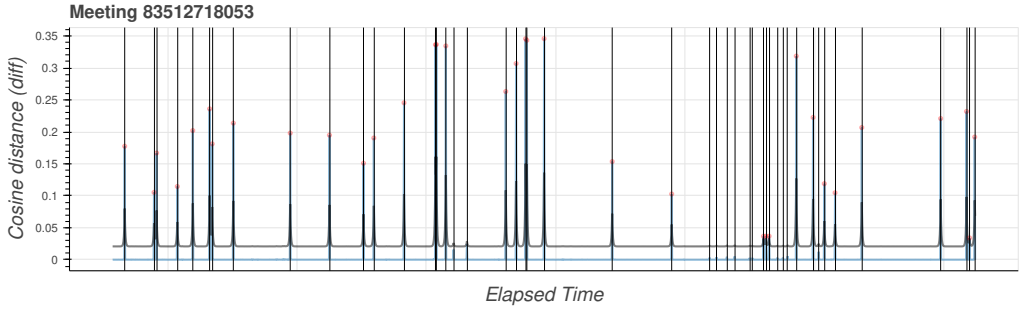
(a)



(b)

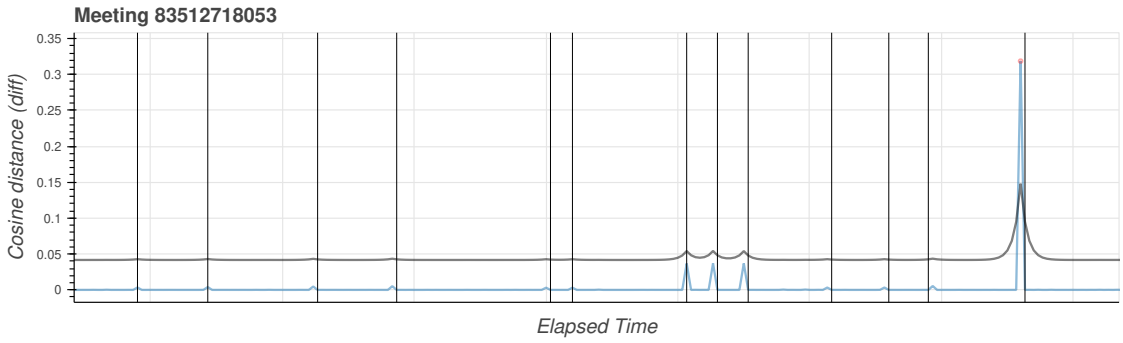


(c)

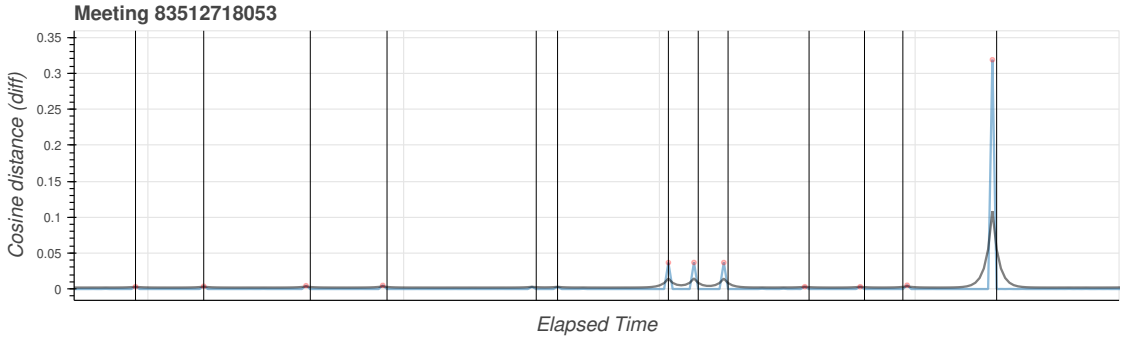


(d)

Figure 3.17: Initial Kalman filter predictions for the four meetings ($pn = 0.5$, $nsig = 1$). The prediction threshold tightens without adding false positives. We can see here that even with a low value of $nsig$, the Kalman filter still maintains a consistent buffer above the data, even in areas of low variance. To see this more clearly, compare meeting 220120 in this figure (c) with the same meeting in Fig. 3.14.



(a) $nsig = 2$



(b) $nsig = 0.1$

Figure 3.18: Kalman filter predictions for an enlarged section of meeting 83512718053. We can use prior domain knowledge to reason about the amount of variation in the data when setting $nsig$. In this case, it was possible to significantly decrease $nsig$ such that very small signals are detected without paying the cost of mispredictions.

very low, so we can safely decrease the value of $nsig$ significantly (Fig. 3.18).

The Kalman filter model was much more powerful than the percentile model, its F1 score differed by an average of 0.005 across the four test videos. This is primarily due to the fact that the Kalman prediction threshold hovered far enough away from our low-variation data such that most signals were detected without sacrificing accuracy. In addition, we only needed to focus on optimizing one parameter (for this domain), which generalized well to the other videos. However, we still needed to optimize a parameter, which could vary for different videos. In our case, the same parameter

State	μ	σ^2
Normal	0.0000453564	0.0000051734
Outlier	0.1850038205	0.0109287335

Table 3.1: HMM state distribution parameters learned for meeting 83512718053.

value worked for all four of the videos. It is not clear if this would hold for a dataset of 100 videos. We also relied on some prior domain knowledge to complete the task. As a result, we will want to explore models that can learn the important characteristics of the data without prior knowledge or feedback from other time series.

3.2.5 HMM-based Model

Given the insights learned from 3.2.2 and 3.2.4, HMMs are a great choice for the next model. They require no prior knowledge, only that the user have some idea of how many hidden states to model. For the time series outlier detection task, we're only worried about two states: the normal state of affairs and the unusual state of affairs. HMMs have been used for this exact task in the past [19] and are efficient at learning hidden state distributions out of the box. The experiments that follow use the **hmmlearn**[3] implementation with default parameters[4] and two hidden states.

The HMM takes a different approach to outlier detection. For the models discussed thus far, the general strategy was essentially to identify datapoints that deviate enough from some main distribution. In the case of the HMM, it assumes each group follows a Gaussian distribution, then learns these distributions for the number of groups it is told to find. This approach is convenient for learning outlier distributions for any given time series and is therefore more generalizable. It is possible to build HMMs with other assumptions on the distributions (eg, exponential). For outlier

[3] `https://github.com/hmmlearn/hmmlearn`

[4] `https://hmmlearn.readthedocs.io/en/latest/api.html#hmmlearn.hmm.GaussianHMM`

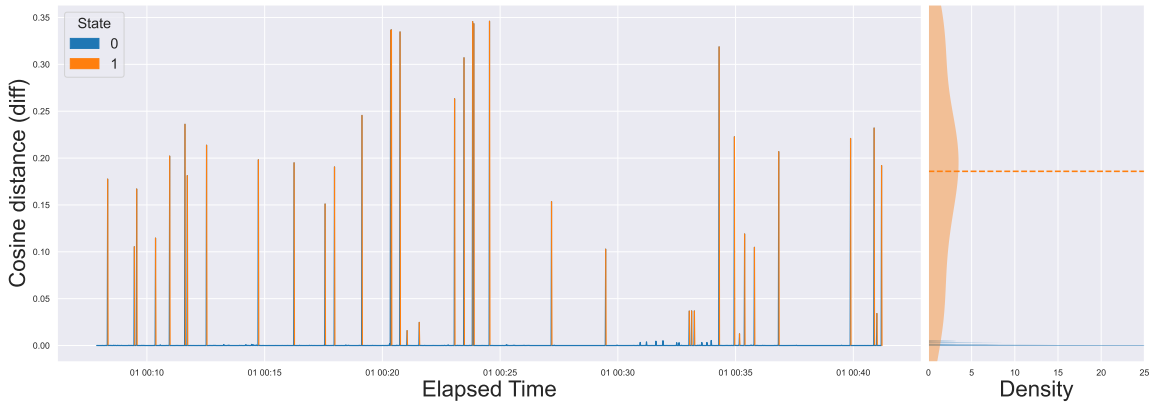
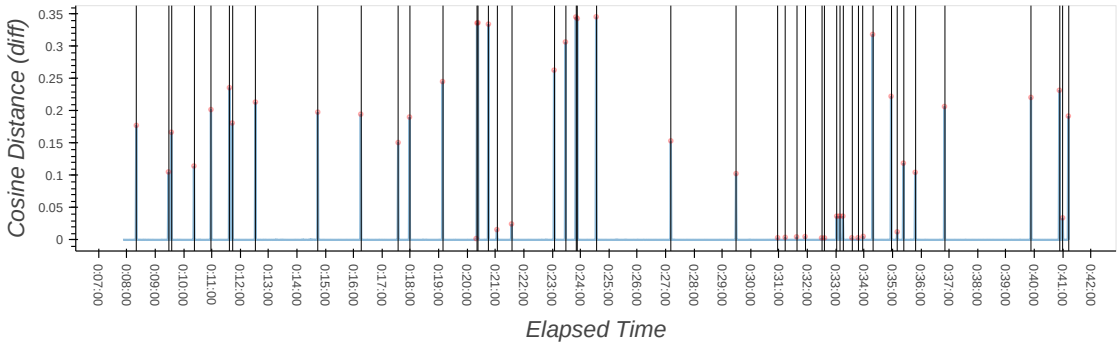


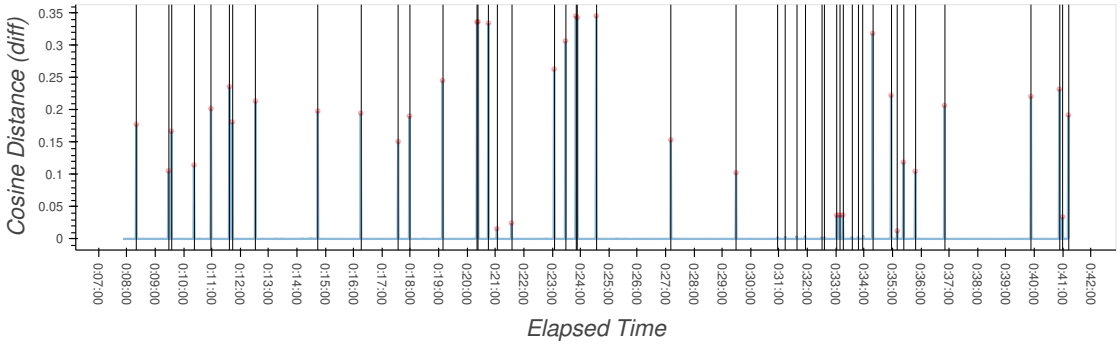
Figure 3.19: HMM state assignments for meeting 83512718053, where the outlier datapoints are assigned to state 1 (orange). Lines take on the color of the datapoint they are connecting to, which produces the orange spikes. The density plot on the right shows the distributions learned from each state, where the orange dotted line denotes the mean of the outlier distribution. Note the x-axis of the density plot. It has a true upper bound of > 5000, therefore needing to zoom in on the range [0, 25] so the outlier distribution can be clearly seen. This follows from the extreme low variance of the blue distribution.

detection, we simply identify the distribution representing the outliers (Fig. 3.19, Table 3.1). Mispredictions occur for datapoints that lie on the boundary between two distributions.

We can see from Figure 3.20 that the HMM does not perform as well as the percentile-based model, with mispredictions occurring for very small signals. Starting at the approximate 31 minute mark, the speaker presented a series of slides where talking points were added incrementally while highlighting results in a table. We can see this more clearly in Figure 3.21. In this case, the only changes in content came from highlighting a different part of the table and adding a bullet point to the talking points. Since the bulk of the content remained constant (title, background, and table), the cosine distance between the respective frames was not large enough for the HMM to detect. This may be improved by fine-tuning the model's parameters such that the distribution belonging to the "normal" hidden state has an extremely small



(a) Optimal percentile-based predictions

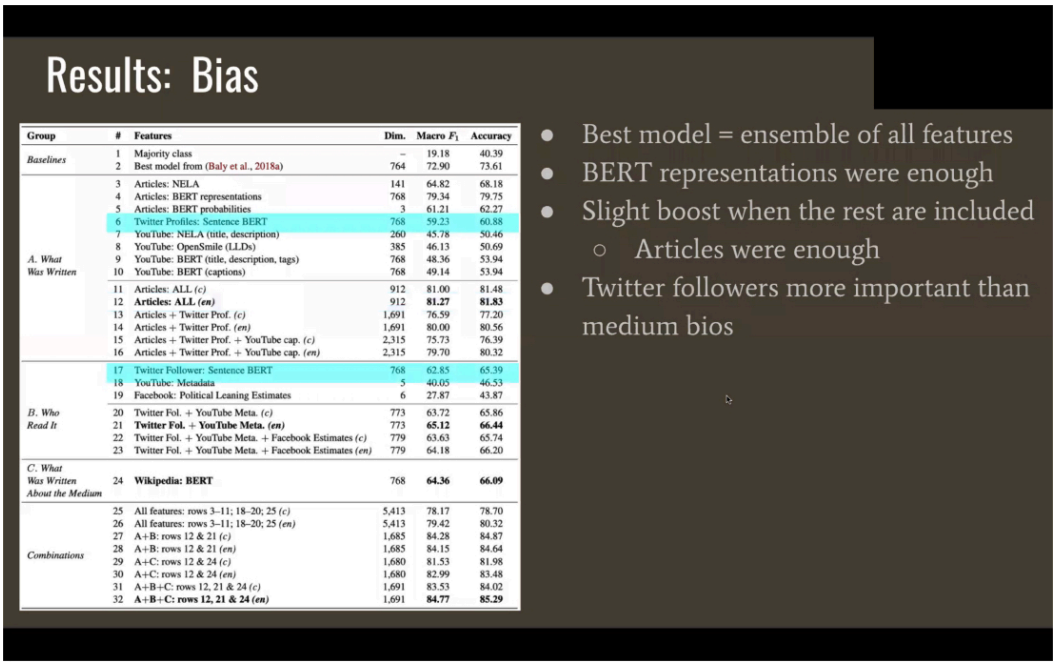


(b) HMM-based predictions

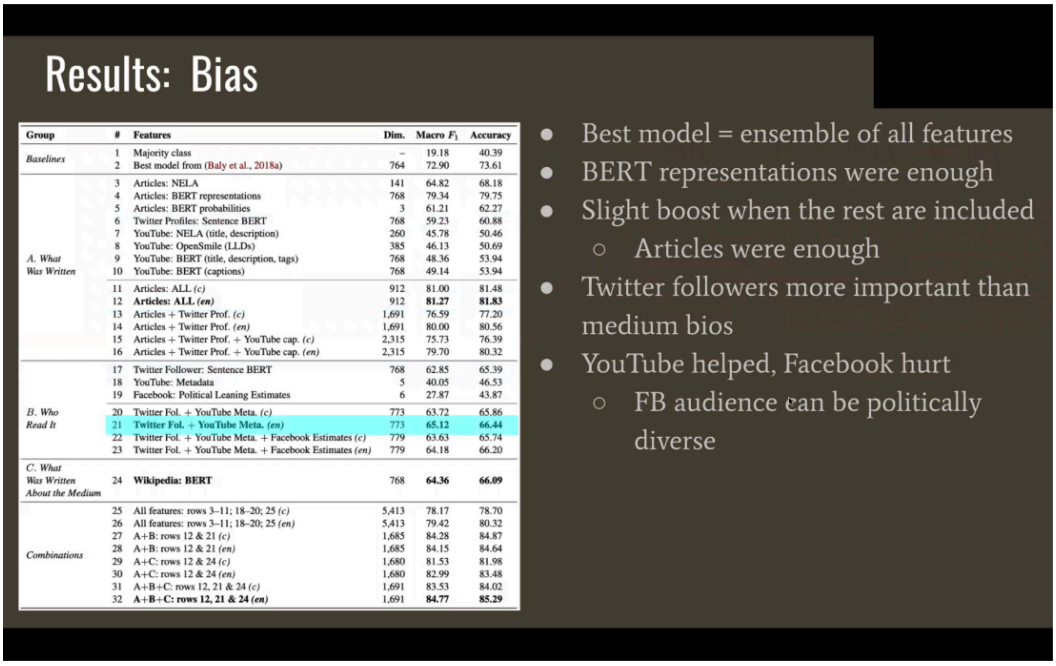
Figure 3.20: Optimal percentile-based predictions vs. HMM-based predictions. The HMM struggles to detect very small signals. See the time interval at approximately 31:00 to 34:00.

variance (even smaller than it already does), which might allow the HMM to place smaller signals in the outlier distribution. While the task of investigating potential improvements is beyond the scope of this thesis, it will make for interesting work in the future.

Though the HMM does not match the performance of the percentile or Kalman models, its F1 scores are within 10% in all cases (Figure 3.22). This deviation is not so unfavorable when considering the ease of use and generalization afforded by HMM. In addition, when compared to the Kalman filter, the HMM is able to handle the



(a) Slide 29



(b) Slide 30

Figure 3.21: An example slide change that was missed by the HMM. In (a), we can see the speaker highlighting results while displaying talking points in the right half of the slide. Then in (b), the highlighting changed and a new bullet point added. Since most of the content remained constant between these two slides, the cosine distance between their respective frames was not large enough to be classified as a slide change.

unmasked videos with consistency while having the potential to improve if slide change definitions are modified. For example, consider the scenario in Figure 3.21 where adding a bullet to a slide produces a very small signal. If the slide change definition were to disallow small changes to slides, which would be favorable for instructors who do not prefer to present content in an incremental fashion, then the HMM's performance would improve. Now that we have a working slide change detector, we can proceed to derive the video metrics.

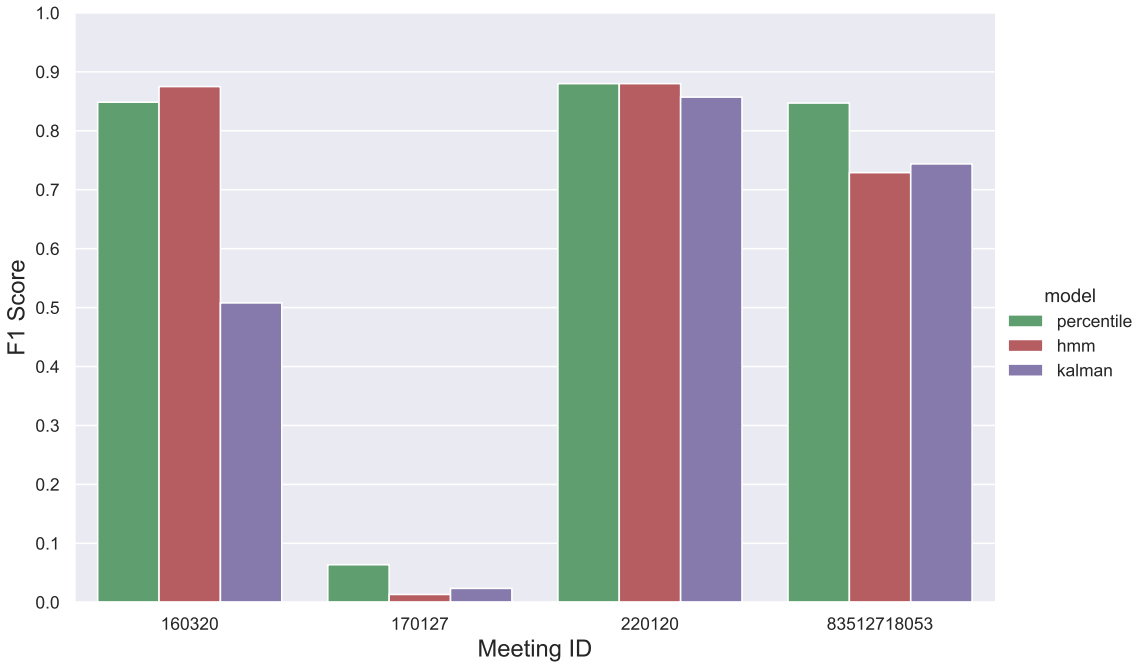
3.3 Video Metrics

3.3.1 Slide Count (SC) and Pace of Slide Change (PSC)

Slide counts are derived by simply counting the number of slide change predictions. Since the PSC derivation depends on slide detection, the following results are sufficient for demonstrating both SC and PSC. In Figure 3.24, we can see mispredictions consistent with those discussed in section 3.2.5. The masked video has legitimate signals, that also happen to be very small and therefore go undetected. The unmasked video has large and noisy signals, which are erroneously picked up by the HMM (see Fig. 3.23).

3.3.2 Words per Slide (WPS)

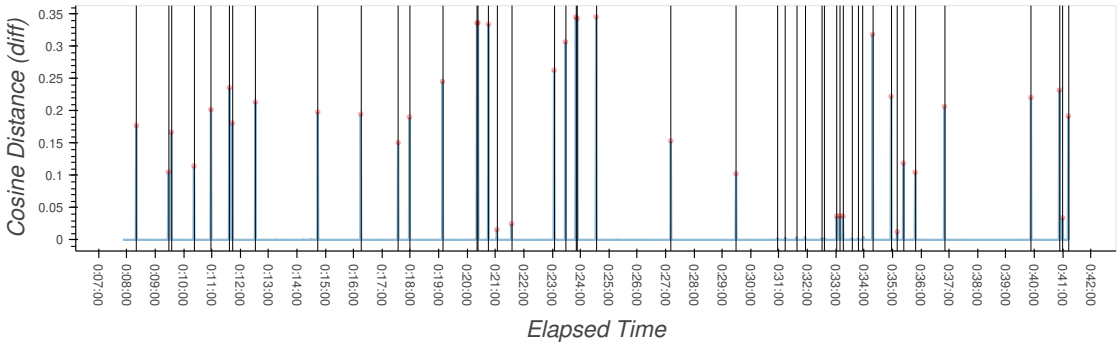
Before WPS can be calculated, the target frames need to be identified and extracted from the video. The alternative would be to run OCR on each frame of the video, then deduplicate. This latter procedure would be extremely inefficient, time consuming, and therefore unpractical for this use case. As a result, we mapped the timestamp of each slide change, or predicted slide change, to the relevant frame number. In theory,



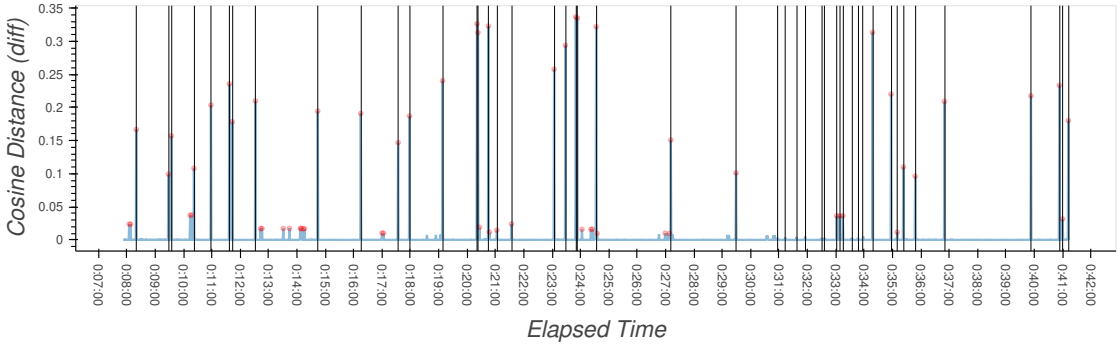
(a) Unmasked

(b) Masked

Figure 3.22: F1 scores for the optimal percentile-based model (green), the HMM-based model (red), and the Kalman filter model (purple). The HMM performs within 10% of the percentile-based model in all cases, which makes the generality gained from the HMM worth the performance cost.



(a) Masked

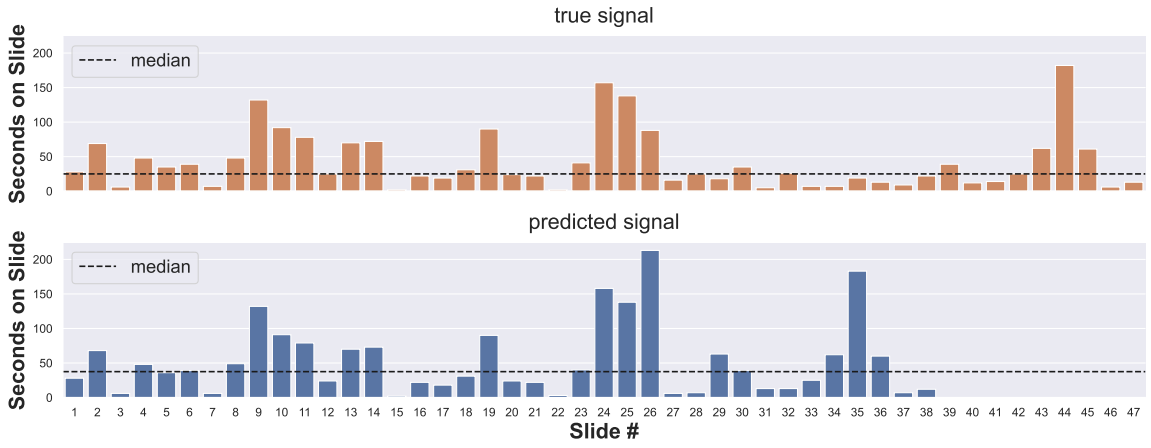


(b) Unmasked

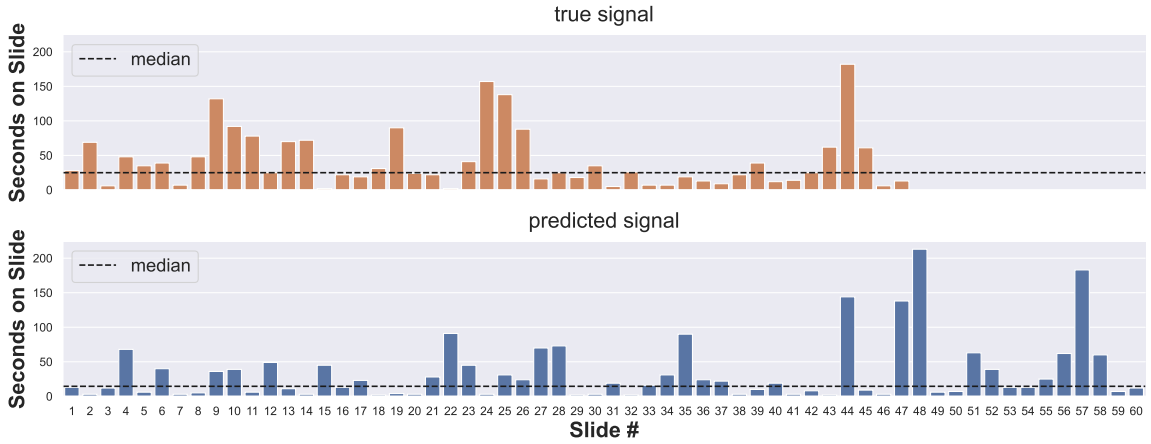
Figure 3.23: HMM predictions for the masked and unmasked cases. This view clarifies why the PSC mispredictions are occurring in Fig. 3.24. In the masked case (a), we can see that small slide change signals are not picked up around the 31:00 mark, which results in an under-counting of the slides (Fig. 3.24a). Conversely, the unmasked case (b) produces extra predictions in the first half of the presentation as a result of the noisy signals produced by the camera. This causes the PSC metric to inflate the slide count (Fig. 3.24b).

this should return a set of unique frames, each representing a different slide.

We started by mapping each frame to its respective timestamp, based on the frame rate of the video, then selected the relevant seconds based on the slide change predictions. Given this set of seconds, we extracted the frame number that had the highest cosine distance value. This process is not entirely error-free since a slide change occurs at the frame-level, while a human annotator is only able to flag it at



(a) Masked

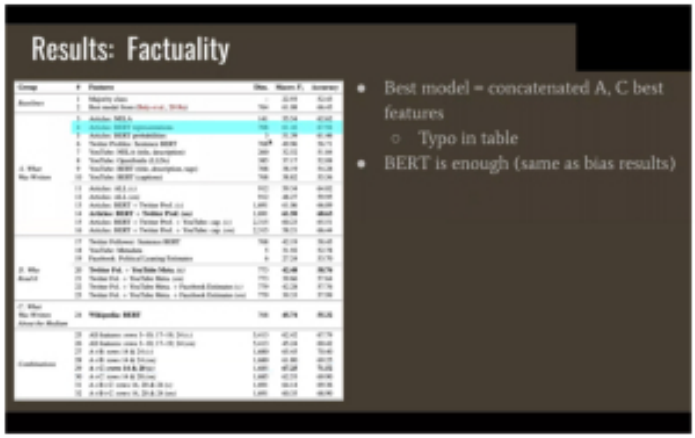


(b) Unmasked

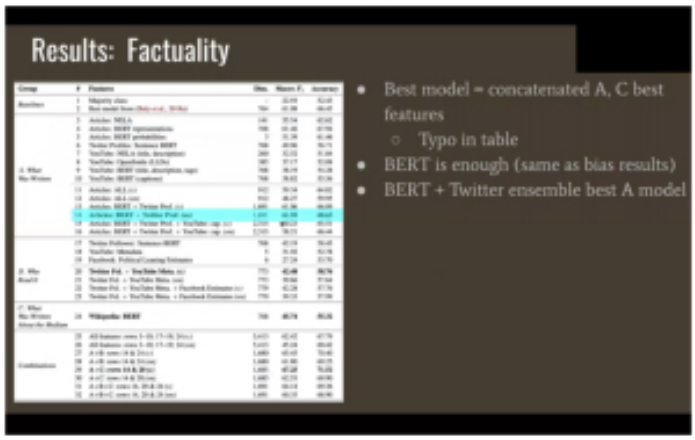
Figure 3.24: HMM predictions for the PSC metric (meeting 83512718053). For the masked case (a), the model missed the small signals towards the second half of the presentation (see Fig. 3.20b). Conversely, for the unmasked case (b), many of the additional noisy signals were picked up by the model, which led to extra slide predictions.

the second-level (Figure 3.25). We can see in 3.26 that the duplicate slides are a result of the slide change happening close to the beginning of a second. Conversely, slide change predictions made by the model will not introduce this type of error since signals at the frame-level are aggregated to the true second where they occur (see Fig. 2.5).

Slide 36



Slide 37



Slide 38

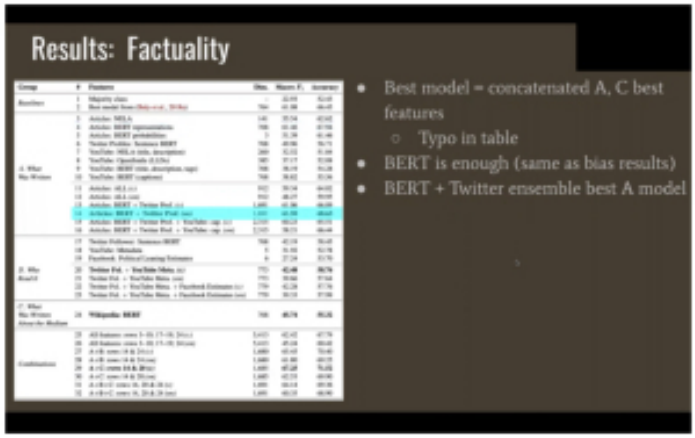


Figure 3.25: A subset of the sequence of frames extracted during the OCR process. In this case, a slight discrepancy in the annotation mapped the 38th slide change to the wrong frame number.

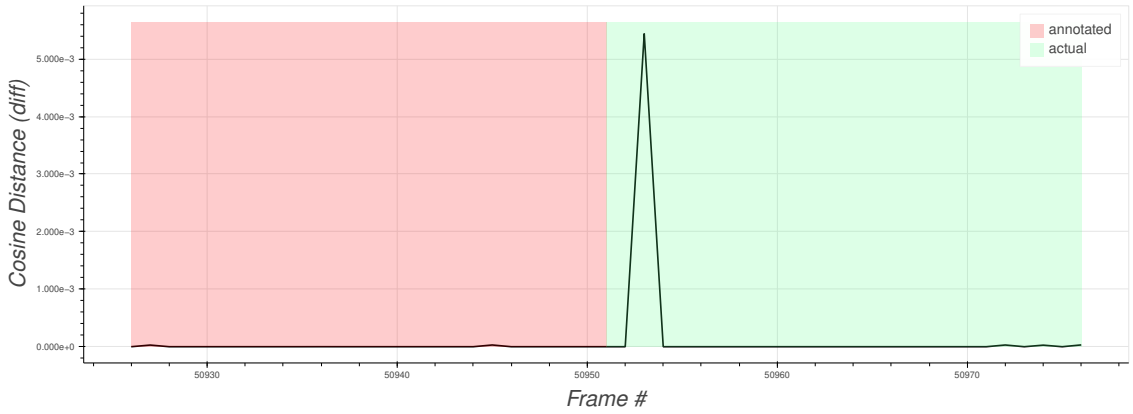
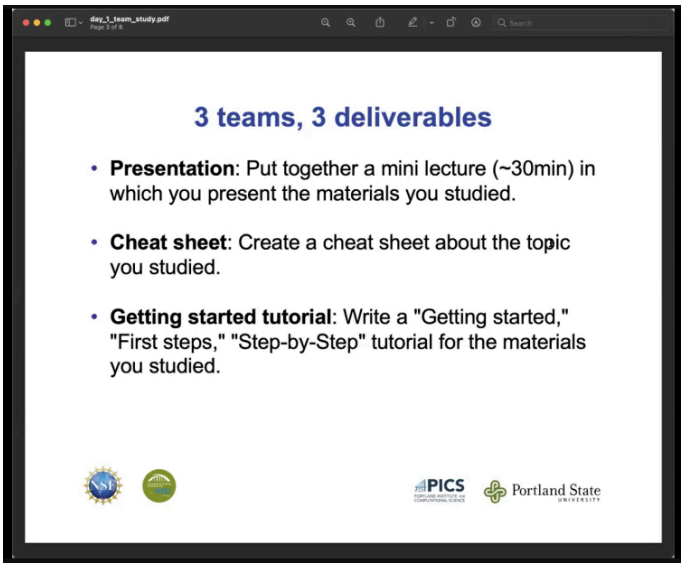


Figure 3.26: A two second period where the seconds are differentiated by the shaded areas. Human annotations will most likely introduce error when the slide change occurs near the transition between seconds. This particular scenario created the duplicate in 3.25. The annotation flagged the slide change as occurring in the period shaded in red, whereas the actual slide change happened in the green area.

Tesseract can be configured in many different ways - there are currently fourteen[5] different page segmentation settings depending on the use case (Figure 3.27). We use the default settings for our purposes, which automatically segments the incoming image. This presents another opportunity for fine-tuning as the default setting may not be appropriate for the task at hand since some presentations may have different text than others.

To calculate WPS, text is extracted from each slide, then tokenized to retain alphanumeric characters. A simple count of the resulting token sequence is the final representation of WPS. We can see from Figure 3.28 that the same misprediction patterns occur with respect to the HMM. In addition, it is clear from meeting 83512718053 that the latter half of the presentation was significantly different from the first half (3.28a).

[5] `https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html`



(a) Sample slide

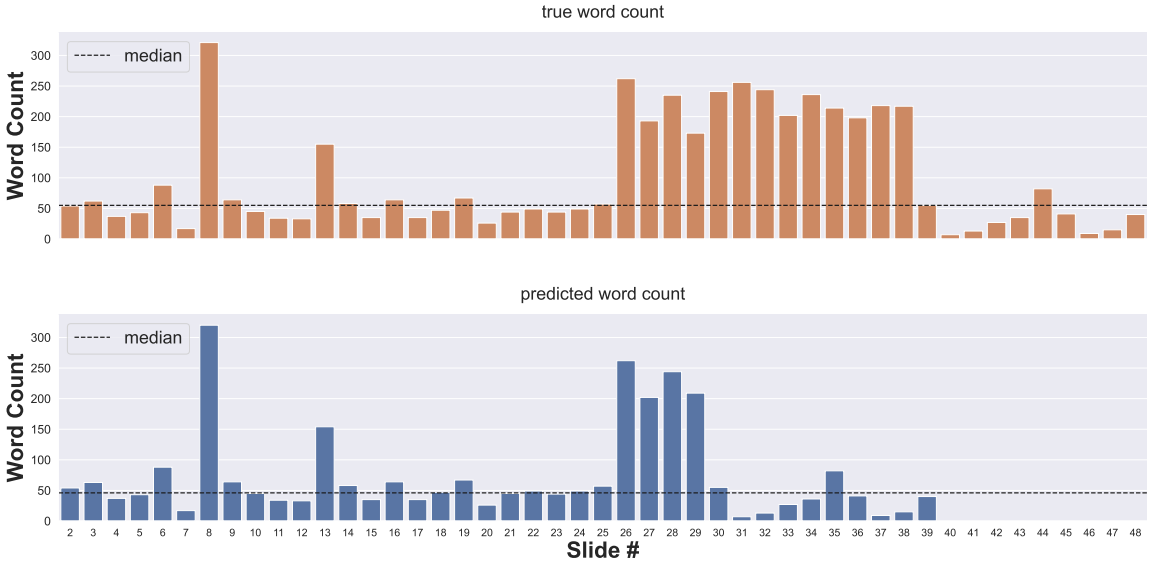
```
we ee
eres

¢ 5s AaAIIVAaArandi
s, 3 deliverak

¢ Presentation: Put together a mini lecture (~30min) in
which you present the materials you studied.

¢ Cheat sheet: Create a cheat sheet about the topic
you studied.

¢ Getting started tutorial: Write a "Getting started,"
"First steps," "Step-by-Step" tutorial for the materials
you studied.
```

(b) Default

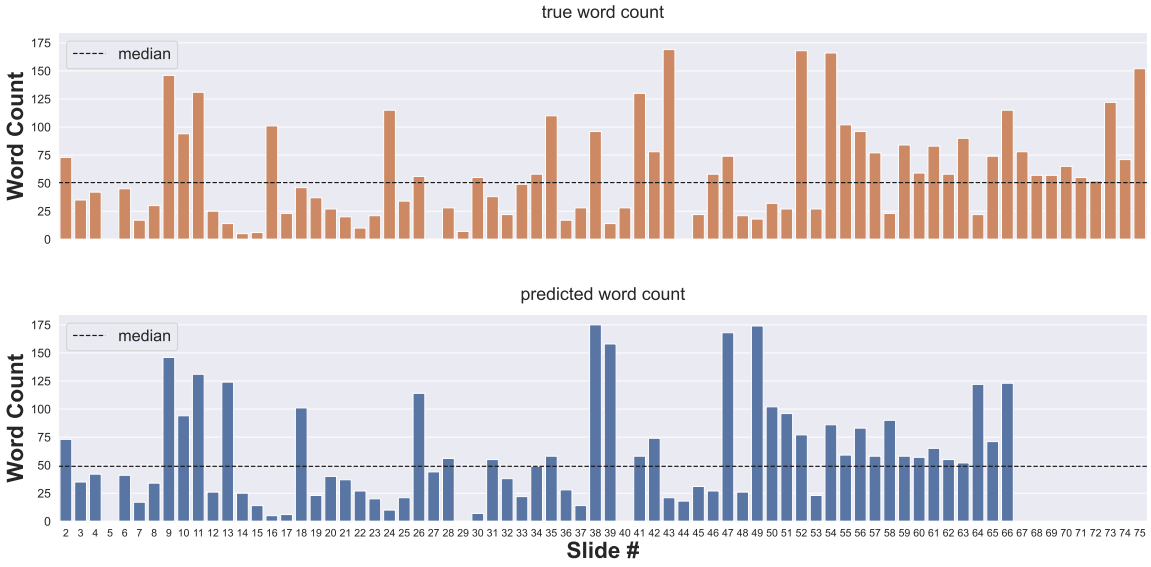
```
Cea aaa CU UE UY = OC
3 teams, 3 deliverables

¢ Presentation: Put together a mini lecture (~30min) in
which you present the materials you studied.

¢ Cheat sheet: Create a cheat sheet about the topic
you studied.

¢ Getting started tutorial: Write a "Getting started,"
"First steps," "Step-by-Step" tutorial for the materials
you studied.

BSS & IPICS GH Portland State
```

(c) PSM 6

Figure 3.27: OCR applied to a sample slide with two different settings. By default, Tesseract attempts to find the block of text withing the image (b), whereas the PSM 6 setting assumes the entire image is composed of a single uniform block of text (c).



(a) Meeting 83512718053



(b) Meeting 170127

Figure 3.28: HMM predictions for the WPS metric (masked case). The variation between predicted and true values is consistent with the HMM behavior in Fig. 3.24a. For both meetings, the HMM underestimated the true slide count. The masked videos have some legitimate signals that are very small, which are not picked up by the model.

3.4 Summary

This chapter evaluated the newly developed Zoom meeting metrics on real Zoom data. First, transcript metrics were compared to those generated by Meeting Measures, thus acting as a validation step. Next, the process of developing a model for detecting slides was outlined in detail, including the challenges encountered along the way. Finally, the HMM ended up being the preferred model for the slide change detector due to the way it generalizes while providing consistent performance. Though it wasn't the top performing model, it was able to stay within 10% of the other F1 scores without requiring any hyperparameter tuning. The model was then applied to our dataset of four videos, where mispredictions occurred when slides changed slightly - such as when a bullet point is added to an existing slide.

4

Conclusion

This work introduced a Python framework for analyzing Zoom meetings for the purpose of improving education delivery. Meeting Measures [2], an R-based tool started in 2020, acted as the baseline for this thesis to build upon. The relevant metrics fall into either the transcript or video categories. The transcript metrics measure the proportion of speaking time per person, the average utterance length per person, speaking pace over time, non-participants, and general sentiment of language. Sentiment scores were derived using open source, pre-trained models from Hugging Face. For validation purposes, our transcript was run through zoomGroupStats [3], the R package powering Meeting Measures. The resulting transcript dataset was successfully validated against the dataset produced by the Python tool.

Video metrics expanded the functionality of the tool beyond what could be learned from transcripts alone. The metrics derived from Zoom videos were slide counts, pace of slide changes, and words per slide. A slide change detector was developed to accomplish these tasks, where four different models were evaluated. Videos were first processed to compute the L2-norm for consecutive frames in order to identify slide change signals. However, slight distortions in many of the frames produced large quantities of false signals. As a result, the cosine distance was used, which mitigated

issues caused by the distortion. Further denoising steps where taken by masking the camera from each frame.

Four different models were then evaluated with respect to their F1 score on a test dataset of four videos. First, a percentile-based model was developed and ultimately performed well. However, the model needed to be optimized for each dataset, which is not ideal in practice. Next, we experimented with a rolling standard deviation model. This option was quickly abandoned as the standard deviation threshold was ineffective for segments of the frame data with extremely low variation. We then built a model based on the Kalman filter, which performed well while being easier to tune. However, this model still required some level of optimization. Finally, we implemented an HMM-based model for its ability to generalize. The HMM did not perform as well as the optimized percentile-based model. It did however score within 10% of the percentile model in all cases. As a result, the HMM was selected to derive the final version of the video metrics.

The final task for developing video metrics was to extract the text from each slide to compute word counts. For this, the open source OCR engine Tesseract was used. We initially noticed some error with the frame extraction procedure. Further investigation revealed the source of the error to be the human annotator. An additional source of error was introduced by the HMM, as it fails to detect slides when the signal is low. Improving model performance is something that could be improved upon in the future.

4.1 Future Work

As larger segments of society transition to remote work and school, it will remain important to have the ability to gather feedback from meeting participants. As a result, this project is something that can continue to be built upon. Better slide detection methods can be explored based on the rules set by the user. For example, frame signals can be amplified for users interested in counting minor changes to slides, such as adding a bullet. This will result in less misses for a model like the HMM. Conversely, frame signals can be further smoothed such that smaller signals are not picked up. In this case, the user would only want to count slide changes when the entire slide changes, instead of just a small portion. Other rules can be set by the user to exclude rapid slide changes in a particular time frame (e.g., no more than 2 slide changes per 10 seconds).

It would also be helpful to experiment with improving the slide detection models such as the HMM or Kalman filter. Additionally, there are deep learning models that might make good candidates, such as the autoencoder. The OCR engine has many options that could be experimented with. It may be the case that certain settings will work better than others, which could improve the results of the slide content analysis.

There are more metrics of interest that can be derived from this data. It would be interesting to formulate a signature for each slide in a presentation for the purpose of tracking the number of times a slide has been revisited. This could help the teacher identify moments in the presentation where students may not have absorbed a concept, thus needing to revisit the same slide multiple times. Student interaction, as determined by the transcript, could be correlated with the slide content to discover patterns responsible for student disengagement. Finally, more intensive video analysis

can be done with some of the other video files available for download. For example, the gallery view can be used to determine how many students have their cameras on to gauge the general level of interest. For those that do have their camera on, models can be developed to detect emotion and other nonverbal cues.

Bibliography

[1] A. Stefanile, "The transition from classroom to zoom and how it has changed education," *Journal of social science research*, vol. 16, pp. 33–40, 2020.

[2] A. Knight, "Meeting measures: Feedback from zoom." `http://apknight.org/meeting-measures-feedback-from-zoom/`, 2020.

[3] A. Knight, "zoomgroupstats." `http://zoomgroupstats.org/index.html`, 2021.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[6] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *arXiv preprint arXiv:1506.06724*, 2015.

[7] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems [j]," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[8] S. Särkkä, *Bayesian filtering and smoothing*. No. 3, Cambridge University Press, 2013.

[9] "simdkalman documentation." https://simdkalman.readthedocs.io/en/latest/, 2021.

[10] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995.

[11] A. Becker, "Kalmanfilter.net." https://www.kalmanfilter.net, 2021.

[12] D. Jurafsky and J. H. Martin, "Speech and language processing (draft)." https://web.stanford.edu/~jurafsky/slp3/, 2021. Material taken from Appendix A.

[13] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[14] "hmmlearn tutorial." https://hmmlearn.readthedocs.io/en/latest/tutorial.html, 2021.

[15] Y. Gong, W. Zhang, Z. Zhang, and Y. Li, "Research and implementation of traffic sign recognition system," in *Wireless Communications, Networking and Applications: Proceedings of WCNA 2014* (Q.-A. Zeng, ed.), vol. 348, p. 553, Springer, 2015.

[16] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual captcha," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1, pp. I–I, IEEE, 2003.

[17] D. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," *arXiv preprint arXiv:1908.10063*, 2019.

[18] A. Rosebrock, "imutils." `https://github.com/PyImageSearch/imutils`, 2021. Published under the PyImageSearch repo.

[19] J. Li, W. Pedrycz, and I. Jamal, "Multivariate time series anomaly detection: A framework of hidden markov models," *Applied Soft Computing*, vol. 60, pp. 229–240, 2017.

Appendix A

Zoom Analysis Code Repository

All code related to the Zoom analysis tools developed for this thesis is publicly available through Github under the MIT open source license at: `https://github.com/cannonja/zoom-analysis-tool`

Appendix B

Sample Transcript

```
WEBVTT

1
00:00:00.299 --> 00:00:00.930
Jack Cannon: Here we go.

2
00:00:09.690 --> 00:00:10.290
Jack Cannon: One second.

3
00:00:13.320 --> 00:00:16.350
Jack Cannon: When I did my presentation, the other day.

4
00:00:17.789 --> 00:00:20.280
Jack Cannon: I got to hit the share button.

5
00:00:22.080 --> 00:00:24.990
Jack Cannon: it's like you click share and then your monitors come up.
```

6

00:00:25.380 —> 00:00:27.120

Jack Cannon: And you sleep like order is.

7

00:00:27.450 —> 00:00:27.780

JoAnna Langberg: yeah.

8

00:00:28.170 —> 00:00:31.500

Jack Cannon: And I expect it to automatically work because that's what it does for teams.

9

00:00:32.790 —> 00:00:34.950

Jack Cannon: Talking they had to interrupt me and go.

10

00:00:35.010 —> 00:00:36.480

JoAnna Langberg: Like whoa whoa whoa.

11

00:00:39.870 —> 00:00:44.700

Jack Cannon: What I need to do is hide floating meeting controls okay.

12

00:00:48.180 —> 00:00:49.170

Jack Cannon: Okay.

13

00:00:52.740 —> 00:00:58.350

Jack Cannon: Okay, so first thing I want to go over some background stuff with you real quick, before I start.

14

00:00:59.400 —> 00:01:02.070

Jack Cannon: So remember how I told you this.

15

00:01:03.480 —> 00:01:11.610

Jack Cannon: I mean, I think I briefly mentioned it, that there's these things that are language models so a lot of what i'm doing now is.

16

00:01:12.330 —> 00:01:19.530

Jack Cannon: is basically we learn all about language models and how they work and how you can use them for lots of different tasks but.

17

00:01:19.860 —> 00:01:32.520

Jack Cannon: Basically, at a high level what a language model does is you can ignore this stuff just know that we're given a set of words, the goal is to.

18

00:01:33.120 —> 00:01:45.990

Jack Cannon: predict what the next word is so it's like the students open there and then you have like this distribution of outcomes laptops books in this case books, has the highest probability, so this model would predict books in the next word.

19

00:01:46.830 —> 00:01:53.070

Jack Cannon: And that's that's basically what language models do and that's kind of like a simplification of it but.

20

00:01:54.420 —> 00:02:04.980

Jack Cannon: You can use these to do all kinds of other language oriented tasks such as turning words into numbers to do useful things with them does that make sense.

21

00:02:06.630 —> 00:02:07.110

Jack Cannon: Okay.

22

00:02:08.670 —> 00:02:10.590

JoAnna Langberg: it's like I don't complete your texting.

23

00:02:11.040 —> 00:02:12.000

Basically.

24

00:02:14.790 —> 00:02:15.810
Jack Cannon: So then here's a.

25
00:02:17.010 —> 00:02:23.040
Jack Cannon: So there's this model called BERT, which is what i'm going to talk about and they use a lot of sesame street names there's a bird there's an elmo.

26
00:02:24.960 —> 00:02:26.220
Jack Cannon: yeah it's pretty funny.

27
00:02:27.930 —> 00:02:45.240
Jack Cannon: So the way they train this BERT model is they trained it on what's called this mask language modeling tasks, where you give the model, a sentence, but you kind of like mask out a few words and its task is to predict what words would fill this in.

28
00:02:47.700 —> 00:02:55.650
Jack Cannon: So then, so that's one thing that it does and and the other thing that does is next sentence prediction, so you put in a word.

29
00:02:56.190 —> 00:03:07.770
Jack Cannon: And then it outputs whether or I guess you put in two senses sentence sentence, a sense be in that tells you if sentence

be is the next sentence from a sentence a.

30

00:03:08.760 —> 00:03:16.920

Jack Cannon: So this super sophisticated model bird was trained using those methods so it's it's they use the ton of data, Google did it.

31

00:03:18.330 —> 00:03:20.280

Jack Cannon: And then you can use this model to do a bunch of.

32

00:03:20.280 —> 00:03:21.030

Jack Cannon: cool stuff.

33

00:03:21.540 —> 00:03:30.960

Jack Cannon: Which is what i'll get into but, but the gist is is that's basically what it does there's another type of model called a GP CT.

34

00:03:31.620 —> 00:03:36.330

Jack Cannon: Where so so one other side is that a language model like are.

35

00:03:36.720 —> 00:03:45.150

Jack Cannon: The ones that are trained with this way, using the mask things, those are mainly used for like understanding text so if you

have a task, and you want to understand what it's saying.

36

00:03:45.600 —> 00:03:55.950

Jack Cannon: or draw any inferences based on what the Texas same use this type of model, but then, if you want to have a Bot that's going to kind of like generate text and, like.

37

00:03:56.400 —> 00:04:06.990

Jack Cannon: generate sentences or take a question in generating answer the question that's another type of model that's not this one, so those are those are kind of the two main tasks for this.

38

00:04:09.510 —> 00:04:12.420

Jack Cannon: So, as far as turning words into numbers.

39

00:04:15.090 —> 00:04:22.800

Jack Cannon: You can use this model to do that and so there's all these different pictures like so it's like once you take a word and you turn it into a number, then you can graph it.

40

00:04:23.400 —> 00:04:29.940

Jack Cannon: And so what you see here is things like relationships between words and how if words are similar.

41

00:04:30.630 —> 00:04:36.330

Jack Cannon: linguistically than they should be similar numbers, they should be similar in whatever space you projected as to.

42

00:04:37.110 —> 00:04:47.220

Jack Cannon: So this one they're just kind of showing you like you have body parts that are all clustered together close to each other, so I would imagine this has like arm leg, you know head.

43

00:04:47.640 —> 00:04:59.730

Jack Cannon: Like those words are all real close together these cities are all real close together and that's that's what should happen he's, since these are similar words, they should be close together when he turned him into numbers to does that make sense.

44

00:05:02.130 —> 00:05:05.460

Jack Cannon: And you can do really cool things with these like you can.

45

00:05:07.200 —> 00:05:22.440

Jack Cannon: Make analogies like, for example, you can do an analogy, you can tell the computer king is the Queen man is to and leave it blank and it'll give you the word woman because that's just the relationship with these words i'll have.

46

00:05:24.240 —> 00:05:30.180

Jack Cannon: So there's all kinds of cool things and then there's other there's Another thing I think is pretty funny so check this out over time.

47

00:05:30.660 —> 00:05:44.790

Jack Cannon: So someone did this over time, like so in the 1900s they did a bunch of they did a word vectors for for 1900s and then you see Okay, this is what gay looks like in 1900 daft flaunting tasteful sweet cheerful.

48

00:05:45.930 —> 00:05:54.360

Jack Cannon: Then in the 1950s it's closer to follow up some witty and bright and then you go into 1900s now gay is more of like a sexuality type of a thing.

49

00:05:54.870 —> 00:06:03.630

Jack Cannon: So you can kind of see how it more throughout time and how you can see, you use these language models to look at these things same with like you know broadcast awful.

50

00:06:06.300 —> 00:06:07.440

Jack Cannon: To understand why I was like.

51

00:06:08.550 —> 00:06:09.240

Jack Cannon: What now.

52

00:06:09.390 —> 00:06:11.280

JoAnna Langberg: You understand why awful majestic.

53

00:06:12.750 —> 00:06:15.990

Jack Cannon: pitches had it it's like Oh, like a different.

54

00:06:16.260 —> 00:06:16.560

JoAnna Langberg: it's like.

55

00:06:18.300 —> 00:06:21.480

JoAnna Langberg: yeah you're falling off and now it's like.

56

00:06:24.780 —> 00:06:31.830

Jack Cannon: So that's kind of like the just you have language models and then you use language models to do things and a lot of the main task is getting.

57

00:06:32.130 —> 00:06:44.100

Jack Cannon: is to represent words as numbers in the most effective way, the better, you can do this, the better your models are going to be able to do things like predict what the next sentences, you know understand sentences that people are saying.

58
00:06:44.430 —> 00:06:53.730
Jack Cannon: read a sentence and be able to tell if, like there's negative emotion or positive emotion stuff like that all those tasks are dependent on how good you can put words into.

59
00:06:54.540 —> 00:07:07.170
Jack Cannon: embedding as they're called and then one final thing i'll say about this is, you can do this with words, but you can also do the senses, you can project entire sentences down into a space like this and you can project entire documents down so you can.

60
00:07:08.460 —> 00:07:22.860
Jack Cannon: The sky's the limit, of course, the longer the document is, the harder it is but there's actually been some pretty good results doing this, so sentences they're real similar to each other linguistically should be putting each other down here, so the same.

61
00:07:23.010 —> 00:07:23.460
Jack Cannon: same thing.

62

00:07:24.420 —> 00:07:26.430

Jack Cannon: Okay that's all the background work.

63

00:07:28.770 —> 00:07:31.020

JoAnna Langberg: i'm an expert I got the lighting fixed.

64

00:07:33.870 —> 00:07:35.160

Jack Cannon: Oh, you did how'd you do that.

65

00:07:35.430 —> 00:07:42.360

JoAnna Langberg: I moved it down because there's a window behind me, so it was taking the bright light of the window is the focus instead of my pale face.

66

00:07:45.930 —> 00:07:54.780

Jack Cannon: Okay, so so now what i'll do is i'll present this research paper this class is a been a lot of research paper reading which I really like.

67

00:07:55.680 —> 00:08:04.800

Jack Cannon: So I think my Professor knows that i'm into like social science applications so she I think she gave me this paper on purpose, which I thought was pretty cool her to do.

68

00:08:06.570 —> 00:08:20.010

Jack Cannon: So with That said, I shall present what was written versus who read it news media profile and using text analysis and social media contracts by Bali at all 2020 prison by jack in was truly.

69

00:08:21.930 —> 00:08:30.660

Jack Cannon: So the motivation for this project is that social media has led to this trust crisis where traditional media are no longer information gatekeepers.

70

00:08:31.890 —> 00:08:40.140

Jack Cannon: Anybody who has an audience can broadcast whatever content they want to their audience and no one's really in control of that anymore like they used to be.

71

00:08:42.990 —> 00:08:52.170

Jack Cannon: And, as a result of that you have fact checkers fact checking organizations who have kind of risen up to address this challenge, so you have.

72

00:08:52.890 —> 00:08:59.460

Jack Cannon: Sites like snopes politico fact etc so that's kind of what they do and they do it manually.

73

00:09:00.120 —> 00:09:08.850

Jack Cannon: Which kind of takes a long time, as you would suggest, as you as you would think, and this is problematic because past research research has shown that.

74

00:09:09.540 —> 00:09:13.380

Jack Cannon: viral claims are usually shared within 10 minutes they've been posted.

75

00:09:13.920 —> 00:09:22.770

Jack Cannon: And then, when something is fake news that spread six times faster than real news would and I guess, not only that, but it reaches far more people as well.

76

00:09:23.550 —> 00:09:34.620

Jack Cannon: So to address this challenge this leads to our research question which is, is it possible to build a real time fact checking system and that's kind of what the authors are looking at here.

77

00:09:35.550 —> 00:09:44.460

Jack Cannon: So some of the assumptions that they're making is they want to profile the medium and not the content and in this context medium is just a news organization.

78

00:09:45.540 —> 00:09:53.460

Jack Cannon: So they use that word throughout the paper and that's what it means in this context, so when they say profile the medium they're basically saying that.

79

00:09:53.850 —> 00:10:03.690

Jack Cannon: They don't really have to worry about analyzing an article or content it's just whoever put publish that content if they.

80

00:10:04.260 —> 00:10:14.310

Jack Cannon: have published a lot of faker bias content in the past they're more likely do it in the future so we'll go ahead and flag that content based on who's publishing it not the content itself.

81

00:10:15.270 —> 00:10:22.710

Jack Cannon: And then the last assumption they make is that bias bias in factuality have both linguistic and social components to it.

82

00:10:23.370 —> 00:10:28.440

Jack Cannon: So that kind of leads to these three pillars of their system which is.

83

00:10:28.860 —> 00:10:39.000
Jack Cannon: They want to look at what was written, which is not just the article is that the medium rights, but all the content they produce so Twitter post YouTube video stuff like that in addition articles.

84
00:10:39.570 —> 00:10:57.360
Jack Cannon: The second one is who read it, which is focusing on their audience the assumption is that if if your audience is is biased one way then you're probably bias to, and the third assumption is what was written about them, so what does Wikipedia right about the particular news organization.

85
00:10:59.250 —> 00:11:06.120
Jack Cannon: So with that said here's their system and i'm going to step into more of these i'm going to step into these, one by one, but basically you can kind of.

86
00:11:06.360 —> 00:11:14.220
Jack Cannon: Tell them basic, just as they have this web scraper that for every news organization they collect articles YouTube videos Twitter profiles.

87
00:11:14.730 —> 00:11:23.430

Jack Cannon: They hit the Facebook marketing API and they get their Wikipedia page they extract a bunch of features a bunch of characteristics, out of it.

88
00:11:23.970 —> 00:11:35.790
Jack Cannon: And then they train a model to predict whether or not they are biased, or whether the stuff they they printed they put out as true or untrue and mixed stuff like that.

89
00:11:37.950 —> 00:11:43.740
Jack Cannon: So let's start with the first one and we'll talk about this, what was written to there and we'll start with the articles.

90
00:11:43.950 —> 00:11:55.440
Jack Cannon: So the first thing they did they do is they extract these linguistic features, so they went and collected a bunch of articles for each news medium extracted bunch of the list of features and they use this news landscape tool kit.

91
00:11:56.190 —> 00:12:11.010
Jack Cannon: From a previous paper and it's not really important to understand or digest everything that's in here, but just know that there's just a bunch of different linguistic features like point of speech or part of speech features linguistic stuff emotion.

92

00:12:12.240 —> 00:12:19.020

Jack Cannon: features bias and subject to be features, so they kind of utilize this tool to extract all of these things for each article.

93

00:12:19.470 —> 00:12:31.140

Jack Cannon: And even though this is at the article level the way they roll this up to the news media level was they just average all of these together to get a representation for for the news media.

94

00:12:33.180 —> 00:12:41.670

Jack Cannon: And then, then they use embedded features which is, which is what we just talked about before the presentation started, and that is, they use this BERT model.

95

00:12:42.210 —> 00:12:46.470

Jack Cannon: And what they did was they find to network model using.

96

00:12:47.490 —> 00:12:58.170

Jack Cannon: An external data set of articles, because they don't want to bias their own work by by fine tinea model on the data set that they're trying to infer on and what they did was they fine tune Burt .

97

00:12:59.520 —> 00:13:14.790
Jack Cannon: To be more specific to their tasks so remember how I said, it is a mass language tasks and the next sentence prediction tasks well the task they're interested in is doing a bias prediction tests, so they kind of fine tune, or to be a bias predictor instead of a mess language model.

98
00:13:16.470 —> 00:13:18.420
Jack Cannon: And they encoded the articles.

99
00:13:19.470 —> 00:13:30.480
Jack Cannon: Using this fine tune model and they they basically took all the articles and projected them down into that mathematical space that that we looked at and they did that but yes.

100
00:13:30.870 —> 00:13:43.770
JoAnna Langberg: So when it was tuned to figure out bias did it look at specific words or was it just because the medium is this or that that's how they that's how I figured it out.

101
00:13:44.640 —> 00:13:49.950
Jack Cannon: So they had a and i'll talk a little bit about the data set after this but they had a data set that.

102

00:13:50.820 —> 00:14:04.590
Jack Cannon: For each article it had a label on whether it was biased or factual so they train the model by giving it the article and then giving it the label and they basically tried to teach the model, how to recognize bias and sexuality.

103
00:14:06.480 —> 00:14:06.630
Jack Cannon: Is.

104
00:14:07.020 —> 00:14:07.890
JoAnna Langberg: going to that later.

105
00:14:08.370 —> 00:14:11.430
Jack Cannon: yeah yeah i'll go into that a little bit later um.

106
00:14:12.540 —> 00:14:13.230
Jack Cannon: So.

107
00:14:13.710 —> 00:14:20.070
Jack Cannon: Also, what they did was one of the limitations in this model is you can only put the five, the first 510 words into it.

108
00:14:21.600 —> 00:14:26.190

Jack Cannon: So, if your article on in most of these articles are going to be a lot longer than 510 so.

109
00:14:27.450 —> 00:14:35.160
Jack Cannon: There could be some limitations there but usually it's a best practice, you can you put the first 510 words into the model.

110
00:14:35.700 —> 00:14:42.810
Jack Cannon: And then you average the word vectors for each of those words to get the the actual decoding for the entire article.

111
00:14:43.800 —> 00:14:50.190
Jack Cannon: And there's a quick aside on that that i'll mention about these first 510 words some previous work has shown that.

112
00:14:50.880 —> 00:14:55.620
Jack Cannon: Comparing two data sets one of them's an imdb data set of movie reviews.

113
00:14:55.950 —> 00:15:03.510
Jack Cannon: And the other ones are Reuters data set, and I think this warders one is news headlines, it might be news articles, but I think it's news headlines.

114

00:15:03.840 —> 00:15:11.430

Jack Cannon: But they're trying to show is that this maximum sequence length is the amount of words that you put into the model before you cut it off.

115

00:15:12.030 —> 00:15:19.170

Jack Cannon: And what they tried this shows that well in the Reuters data, so if you cut down that maximum sequence length, then.

116

00:15:19.860 —> 00:15:24.900

Jack Cannon: Then what this is on the y axis is the performance of the model how accurate, it was basically.

117

00:15:25.470 —> 00:15:35.640

Jack Cannon: It doesn't suffer very much it suffers a little bit you know, and these are the blue and the orange are two different models so both models didn't suffer very much from here to here.

118

00:15:36.060 —> 00:15:45.150

Jack Cannon: But from here to here they had a little bit of a dip but not very much whereas on the imdb data set if you reduce the amount of see.

119

00:15:45.540 —> 00:16:04.260
Jack Cannon: sequence length, it takes quite a hit like this is like six points from here to here and then like another six let's see 54 to 50, so this is like four points, and this is like six points, so the authors of this previous paper thought that these imdb our reviews are pretty long.

120
00:16:05.310 —> 00:16:14.970
Jack Cannon: That on average there and much longer than the than the documents that were in this Reuters data set so this personalization was much more severe that's just an aside on that.

121
00:16:16.560 —> 00:16:32.340
Jack Cannon: Then, the last thing they did was they calculated these posterior probability vectors for each news article and the way they did that was They ran our their articles through this bird model, the same way they did in the previous step with articles.

122
00:16:33.690 —> 00:16:51.540
Jack Cannon: But instead they converted the output of that to a probability, so in this case in this example, there are three bias labels and three factuality labels which i'll go into later, but essentially the bias is like left bias Center bias right bias and the functionality is.

123

00:16:53.820 —> 00:17:00.060
Jack Cannon: it's like high factuality mixture and low factuality or something like that.

124
00:17:00.900 —> 00:17:01.530
JoAnna Langberg: or something.

125
00:17:02.010 —> 00:17:11.910
Jack Cannon: yeah basically so this probability the way you did you to you if you put an article in and you got these vectors out the way you interpret that is is that it's most likely a high bias.

126
00:17:12.090 —> 00:17:16.590
Jack Cannon: or i'm sorry a writing bias, because this the right leaning part has the highest.

127
00:17:16.650 —> 00:17:33.990
Jack Cannon: Has the highest number, and that that they're most likely a factual because the biggest number here is in the factual slot so that's basically what they did so they combine the linguistic features, the embedded features in these prediction vectors and that 's how they included the articles.

128
00:17:35.730 —> 00:17:42.630

Jack Cannon: Secondly, now we move on to YouTube videos they did the same thing with the linguistic features they extracted them.

129

00:17:43.470 —> 00:17:58.050

Jack Cannon: From the news sites title description caption and tags from the video, and then they also embedded those features by running the description in the title through that Burt model and embed that in encoded that.

130

00:17:59.970 —> 00:18:11.460

Jack Cannon: This last part I thought was really interesting, they also took the audio from these YouTube videos and they process the speech or the audio and they extracted a bunch of these acoustic features they call them low level descriptors.

131

00:18:11.790 —> 00:18:24.300

Jack Cannon: From a previous research paper they implemented this open smile toolkit So these are a bunch of acoustic features loudness single energy pitch you know blah blah blah stuff like that so, then they.

132

00:18:24.960 —> 00:18:31.710

Jack Cannon: They thought that this would be good, because these features have been shown to be useful for like a motion detection and speech I guess that.

133

00:18:31.890 —> 00:18:35.010

Jack Cannon: kind of makes sense because our did you did you have a question.

134

00:18:36.300 —> 00:18:47.010

JoAnna Langberg: or a comment as well, like that makes sense because, like a lot of times you'll have a news anchor and they're all very monotone you're like on today that it ended up whenever you have like a.

135

00:18:47.610 —> 00:18:53.370

JoAnna Langberg: tea spill kind of a YouTube video it's always like then this happened, blah blah blah it's like very.

136

00:18:54.120 —> 00:18:54.330

Jack Cannon: yeah.

137

00:18:54.360 —> 00:18:55.410

JoAnna Langberg: all over the place.

138

00:18:55.860 —> 00:19:00.720

Jack Cannon: yep exactly like a lot of times they're more snarky you know so.

139
00:19:01.860 —> 00:19:03.810
Jack Cannon: An angry quite free.

140
00:19:04.050 —> 00:19:05.490
JoAnna Langberg: More emotional yeah.

141
00:19:05.520 —> 00:19:06.240
For sure.

142
00:19:09.000 —> 00:19:18.600
Jack Cannon: Okay, so those were the is so the final part about that first pillar of what was written what kind of content they produced they went and got their Twitter profiles.

143
00:19:19.110 —> 00:19:31.350
Jack Cannon: And what they did with these Twitter profiles was the encoded these with s Burt instead of birth, an expert stands for sentence Burt it's a variation of birth it's kind of more designed for embedding sentences for the most part.

144

00:19:31.800 —> 00:19:36.120
Jack Cannon: And they they kind of they had two reasons for using this and that is that.

145
00:19:36.930 —> 00:19:38.820
Jack Cannon: The Twitter profiles that they scraped.

146
00:19:39.120 —> 00:19:46.980
Jack Cannon: They didn't really have enough data to fine tune Burke, because with these machine learning models, you really need a lot of data that's kind of what makes them powerful because the more data, you have the better.

147
00:19:47.310 —> 00:19:50.070
Jack Cannon: So they didn't really have enough data to fine tune it.

148
00:19:50.370 —> 00:20:03.600
Jack Cannon: So, and then the second thing was that Twitter profiles kind of kind of more they look more like sentences than they do documents or articles, so they use the sentence model, specifically for for encoding Twitter profile descriptions.

149
00:20:04.140 —> 00:20:15.150

Jack Cannon: And the other thing they did was they they got metadata such as well, they calculated statistics from these Twitter profiles like are they verified, you know what geographical region number of followers so on and so forth.

150
00:20:15.720 —> 00:20:20.760
Jack Cannon: And then, for any news medium that didn't have a Twitter profile, they just got all zeros everything was zero now.

151
00:20:22.020 —> 00:20:25.980
JoAnna Langberg: But it include like emojis or was it just words.

152
00:20:26.370 —> 00:20:42.960
Jack Cannon: um I don't know I mean they basically I don't think it included a mo geez I think they they did like six or seven of them, and it was just number of followers maybe number of retweets or something like that, but emojis might have been an interesting thing to capture for sure.

153
00:20:46.620 —> 00:20:49.290
Jack Cannon: That we move on to the second pillar, about making.

154
00:20:50.970 —> 00:21:03.270

Jack Cannon: inferences about the audience and the key idea is that your audience is indicative of your political leanings so maybe this will, this will help give more information about predicting whether or not a medium is biased in our factual.

155
00:21:04.320 —> 00:21:04.830
Jack Cannon: So.

156
00:21:06.360 —> 00:21:16.200
Jack Cannon: What they did was for each news proof news medium they went and found 5000 Twitter followers for each one and they went, and they got all their profiles.

157
00:21:16.560 —> 00:21:33.360
Jack Cannon: And they embedded their profile descriptions with expert the same way that they did in the previous slide with the the other Twitter data and then for all 5000 in coatings than the average those together to get a single embedding for the entire medium.

158
00:21:35.010 —> 00:21:43.830
Jack Cannon: Now, the second part was interesting I never heard this before, but there's a Facebook marketing API and.

159
00:21:45.030 —> 00:21:56.070

Jack Cannon: So it's not a surprise that Facebook knows everything about us and they have a profile for all all everybody everybody who's a Facebook user has a profile they've put together profile.

160

00:21:56.580 —> 00:22:09.900

Jack Cannon: And they've absolutely profiled which side of the political spectrum you're on, so what they did was given a particular ID for news medium they call it interest ID I guess it's because.

161

00:22:10.680 —> 00:22:15.390

Jack Cannon: You can pick things as your things that you're interested in on Facebook, so if.

162

00:22:16.080 —> 00:22:26.430

Jack Cannon: For example, if you have an if you have the ID for fox news, you can query the Facebook marketing API and you can get all the users who are interested in fox news.

163

00:22:26.880 —> 00:22:32.160

Jack Cannon: So, not only do you get the users, but then you get where Facebook thinks they are in the political spectrum.

164

00:22:32.640 —> 00:22:37.890

Jack Cannon: So they bring back about seven buckets and It ranges from very liberal are very conservative.

165

00:22:38.550 —> 00:22:48.330

Jack Cannon: So once they've done that, for each news media once they've received the results of this query and they have this set of users and their political spectrums their political orientation.

166

00:22:48.690 —> 00:23:03.660

Jack Cannon: They extract the distribution of the political orientation from the audience, so you can kind of tell in general, where the audience is on the on the spectrum and that that was the feature for this Facebook interest ID.

167

00:23:05.850 —> 00:23:11.490

Jack Cannon: So it's not called out on the paper for looking at audience is more YouTube stuff.

168

00:23:11.850 —> 00:23:22.050

Jack Cannon: So it's not displayed in the system diagram but they extracted a bunch of general audience interaction stats just number of likes number of views dislikes comments number of comments, etc.

169

00:23:22.620 —> 00:23:28.530

Jack Cannon: And then they average those out as well to get the news level representation.

170
00:23:29.310 —> 00:23:31.770
Jack Cannon: So we go to the final part what was written about him.

171
00:23:32.130 —> 00:23:43.800
Jack Cannon: And this was just for each news medium they got their Wikipedia page and they encoded the Wikipedia page the exact same way they did with the news articles, so they use the fine tune Burt model and encourage them that way.

172
00:23:44.220 —> 00:23:50.520
Jack Cannon: And the so for mediums that didn't have a Wikipedia page, they just got it got it out same with the Twitter.

173
00:23:52.080 —> 00:23:53.310
Jack Cannon: So that yes.

174
00:23:54.150 —> 00:24:02.880
JoAnna Langberg: See not we already mentioned that they also go to like that news media's website or was it just a Wikipedia page.

175

00:24:03.240 —> 00:24:04.650
Jack Cannon: Just the Wikipedia page.

176
00:24:05.940 —> 00:24:22.230
Jack Cannon: yeah so so that's a good question so they the, the purpose of the Wikipedia was to see what's written about them on Wikipedia and to see what the medium thinks about themselves, they use their Twitter profiles and then their YouTube channel descriptions for that.

177
00:24:23.850 —> 00:24:31.680
Jack Cannon: yeah but I probably what you're thinking is that well, what does the websites description say about who they are, and what your values are yeah exactly.

178
00:24:34.320 —> 00:24:39.930
Jack Cannon: um so this kind of goes back to the question you had previously about what data they were using and how they were using it.

179
00:24:40.410 —> 00:24:46.980
Jack Cannon: So there's this website it's a media bias fact check website and what they do is they have about.

180

00:24:47.460 —> 00:24:57.960
Jack Cannon: Right now, I checked the other day, and it was like 3100 news organizations that they've profiled on their website and what they do is for each news organization.

181
00:24:58.320 —> 00:25:09.450
Jack Cannon: They give them a bias ranking and affection quality ranking and there's actually more book it's in this right here like on the political bias, they give them I think it's.

182
00:25:10.740 —> 00:25:20.520
Jack Cannon: Left left Center Center right Center in right, I think, is what they do, but the authors looked at a lot of those like.

183
00:25:21.450 —> 00:25:26.340
Jack Cannon: In between buckets and thought that they might have been a little ill defined so they dropped them.

184
00:25:26.820 —> 00:25:39.720
Jack Cannon: And then, they also chose the merge the categories like left Center and left, they just put them into the Left bucket so they kind of made some executive decision on what they did with the labels there, and on the factuality.

185

00:25:41.880 —> 00:25:51.930
Jack Cannon: I think these three were always there low mixed in high so it's like mixed is like sometimes they factual sometimes they're not .

186
00:25:53.670 —> 00:26:04.770
Jack Cannon: What it was all said and done, they they ended up with 864 news mediums and you can kind of tell by eyeballing this that they're not representative Lee.

187
00:26:05.400 —> 00:26:16.350
Jack Cannon: they're, not even the representative between the buckets so it's it's an those are in balance classes, as we say in machine learning so that that come that's going to come into play later.

188
00:26:16.920 —> 00:26:22.650
Jack Cannon: The other interesting thing is a these data sources and I just went over Wikipedia pages Twitter blah blah blah.

189
00:26:23.640 —> 00:26:33.060
Jack Cannon: These were the this was the percent complete this they were able to get for these news mediums so it's like of all the news mediums of these 864 news mediums they only.

190

00:26:33.720 —> 00:26:45.810
Jack Cannon: Only half of them had YouTube pages only 60% of them had a Facebook interest ID so on and so forth, so this was kind of incomplete here only 60% of them had a Wikipedia page.

191
00:26:46.350 —> 00:26:51.210
JoAnna Langberg: So that's what that means so it's like 50% of them had a YouTube page or 15 or like.

192
00:26:52.080 —> 00:26:59.010
JoAnna Langberg: Out of total the total numbers like 50% completed like they had all the information in their YouTube page.

193
00:26:59.550 —> 00:27:02.970
Jack Cannon: Oh no it's 50% of them had a YouTube page.

194
00:27:03.360 —> 00:27:07.440
JoAnna Langberg: Okay, so it's like percentage of the total number of mediums.

195
00:27:07.800 —> 00:27:10.020
JoAnna Langberg: Yes, okay yeah.

196

00:27:12.660 —> 00:27:14.820
Jack Cannon: So the experimental setup was they.

197
00:27:14.970 —> 00:27:25.290
Jack Cannon: extracted all these features that we just went over in detail and given those features they trained a model to predict whether or not that particular news medium.

198
00:27:25.920 —> 00:27:34.320
Jack Cannon: They you know to test the first task is how by a star they give a bias prediction, the second task is, you know how effectual are they.

199
00:27:34.950 —> 00:27:48.150
Jack Cannon: Give a prediction for that so they did a appalachian study which i'll go over in the next few slides but all that is is you train a bunch of components to your system individually.

200
00:27:48.570 —> 00:27:58.440
Jack Cannon: So you can see the contribution that each part of the system has and because they did that ended up with like 60 models so they they trained, a lot of models.

201
00:27:59.040 —> 00:28:06.540

Jack Cannon: And the the model that they chose to use the the the classes that class of model, the type of model was asked me to support vector machine.

202
00:28:07.350 —> 00:28:15.630
Jack Cannon: And the way they did it was they did a five fold cross validation procedure and tune their hyper parameters on the grid search that don't really worry too much about that.

203
00:28:17.010 —> 00:28:32.460
Jack Cannon: And they evaluated the models using this macro F1 score, which is kind of an average of how many correct so it's like of the different classes, the computer team, like, for example, um.

204
00:28:34.350 —> 00:28:41.820
Jack Cannon: But you know left by a Center bias right bias, how many of the right biased truly right bias ones, did you get right.

205
00:28:43.740 —> 00:28:47.370
Jack Cannon: versus how many of the.

206
00:28:49.020 —> 00:28:52.470
Jack Cannon: Of all the errors so it's hard to explain just basically like.

207

00:28:54.870 —> 00:29:10.920

Jack Cannon: How precise were you and then of the things that you got wrong where you more wrong on certain classes and you were on other classes, so this just kind of averages up that's what this macro point score is your kind of gives a full representation on how well the model performed.

208

00:29:11.670 —> 00:29:27.060

Jack Cannon: And the reason why you want to use F1 score is because the classes were in balance, like I showed on the left, look last slide if there were the exact same number in each category, then you could just use a straight up accuracy percent correct, and that would be it.

209

00:29:30.120 —> 00:29:39.540

Jack Cannon: So we'll start with the results with the bias prediction, so this is, this is what I meant by aberration study each row is a model.

210

00:29:40.080 —> 00:29:54.180

Jack Cannon: That got trained with a certain set of features and then the performance was evaluated on that set of features, so you have you know the articles YouTube videos, then you have all of those things combined.

211

00:29:54.570 —> 00:30:03.120

Jack Cannon: Various combinations of them so on so forth, and this I just learned what an aberration study was when I was reading this paper and it comes from.

212

00:30:04.080 —> 00:30:12.450

Jack Cannon: I think neuroscience where when researchers were trying to understand how the brain works, they knew that there were like different subsystems of the brain.

213

00:30:12.750 —> 00:30:20.160

Jack Cannon: So what they would do is they would like remove one subsystem and see how that impacts, the brain like they would do that on the animals basically.

214

00:30:20.700 —> 00:30:33.810

Jack Cannon: and see how the brain performed when it was missing this particular subsection and they would do all kinds of combinations on that to try to get a picture on how the brain works which I kind of cringe a little bit when I when I read that.

215

00:30:34.530 —> 00:30:35.460

JoAnna Langberg: The call.

216

00:30:35.550 —> 00:30:36.780

Jack Cannon: yeah yeah.

217

00:30:38.490 —> 00:30:48.030

Jack Cannon: So I guess the the the best model in this bias prediction is all the features from a, which was the what was written all the features from the.

218

00:30:48.300 —> 00:30:50.040

Jack Cannon: And with a PDF features.

219

00:30:50.130 —> 00:30:50.910

Jack Cannon: All together.

220

00:30:51.930 —> 00:30:56.910

Jack Cannon: It turns out that that got this 85% accuracy type of a score.

221

00:30:58.860 —> 00:31:11.970

Jack Cannon: But there are some interesting things to look at like when when they took the features of just taking the articles and encoding them with Burt just doing that got them like 79% so that actually

performed pretty good.

222

00:31:14.160 —> 00:31:25.200

Jack Cannon: um, whereas the best model here in this a section was all of the features combined together got you at 1% So if you really cared about doing the extra work.

223

00:31:25.860 —> 00:31:36.870

Jack Cannon: You could bump your accuracy from 79 to 81 but really the takeaway from this is that the advert representations, with just the articles are good enough you didn't even need to mess with YouTube or Twitter.

224

00:31:39.210 —> 00:31:41.310

Jack Cannon: The Twitter followers were more important.

225

00:31:42.870 —> 00:31:54.690

Jack Cannon: than the news mediums own bio which that kind of confirms the hypothesis that you know the bias of your audience will be indicative of your bias.

226

00:31:57.210 —> 00:31:59.070

Jack Cannon: And YouTube helped a little bit.

227

00:32:00.450 —> 00:32:12.540

Jack Cannon: When it came to analyzing the audience stuff YouTube helped , but Facebook actually hurt hurt it so Facebook turned out not to really be an important predictor here.

228

00:32:13.350 —> 00:32:22.770

Jack Cannon: except for the fact that you know I think they thought that your Facebook audience could be more politically diverse than you would think.

229

00:32:23.280 —> 00:32:35.910

Jack Cannon: So that may not be the best signal for whether or not your bias just by looking at the the distribution of your Facebook audience that's kind of what they thought and then Wikipedia just straight up just perform poorly 64%.

230

00:32:37.830 —> 00:32:49.560

Jack Cannon: And, in general, the poor coverage is kind of where I mentioned the last slide in general, like the Wikipedia the YouTube Twitter like they weren't able to collect data for all of the news medium, so if.

231

00:32:50.100 —> 00:33:01.290

Jack Cannon: In theory, I guess, if they were able to do that maybe maybe some of these results will be a little bit better, but right now that's probably that could be what what is making these these predictions so bad.

232
00:33:03.870 —> 00:33:15.210
Jack Cannon: So now, if you look at the factuality prediction, you can tell right off the BAT like i'll go back to this slide like you know you have some pretty high number 7981 and then the best model was like 84.

233
00:33:15.630 —> 00:33:32.970
Jack Cannon: But then, if you look at factuality The numbers are a lot lower like 60 4067 their best model got 67 and that was a combination of the a and the sea features so Wikipedia plus what was written.

234
00:33:35.400 —> 00:33:40.860
Jack Cannon: In the same is true as as in the last task is up here like the.

235
00:33:41.640 —> 00:33:57.720
Jack Cannon: representations from the articles were enough like he didn't really need to do very much else, adding the rest of the the data only bumped you from 61.46 61.5 so it's like not even worth wasting

your time going through YouTube videos and Twitter profiles and so on and so forth.

236
00:33:59.550 —> 00:34:18.090
Jack Cannon: And here for the Twitter, the results were reversed from the last one in the last one, the Twitter followers give you a higher score, but in this one, the the Twitter profile of the medium itself gives you a higher score so for factuality This gave a better signal than the followers did.

237
00:34:19.470 —> 00:34:20.640
Jack Cannon: And that was because.

238
00:34:22.410 —> 00:34:29.610
Jack Cannon: yeah you know I mentioned that the results were pretty bad in this factuality tasks and the authors kind of hit the nail on the head.

239
00:34:30.330 —> 00:34:40.710
Jack Cannon: When they are kind of analyzing this and basically was just like look if you're going to predict facts well, you need to have some sort of external data source or knowledge base like.

240
00:34:41.310 —> 00:34:45.060

Jack Cannon: You can't analyze the linguistic structures of sentences or .

241
00:34:45.600 —> 00:34:57.750
Jack Cannon: The content of your audience like in order to actually know if something's true you actually have to go verify at the source that it's true and that's a hard thing to do so that's kind of why they made this claim and I actually agree with that.

242
00:34:58.920 —> 00:35:09.630
Jack Cannon: So the conclusion after all is said and done, the thing that was most important is what they actually write the content they produced, and specifically the articles that they write.

243
00:35:10.650 —> 00:35:23.040
Jack Cannon: But if you add context from social media, it will help a little bit, but not much, and again if they weren't able to collect more complete data and get more social media context, maybe that would help.

244
00:35:24.600 —> 00:35:32.190
Jack Cannon: So the strengths, I thought that they did some very creative feature engineering like specifically with extracting the audio features from YouTube.

245

00:35:32.610 —> 00:35:47.700

Jack Cannon: That was pretty cool system diagram I thought was easy to understand, they seem to do a very thorough analysis of their results, and they were not shy about you know admitting what the limitations of their of their system was.

246

00:35:49.500 —> 00:35:56.940

Jack Cannon: And so now i'm going to give some of my thoughts about this i'm not going to call out any weaknesses, because.

247

00:35:57.450 —> 00:36:07.200

Jack Cannon: This is more of like a subjective thing, and since they're applying their system to social sciences, a lot of the key assumptions they make are subjective so, for example.

248

00:36:07.860 —> 00:36:14.850

Jack Cannon: I don't think you should use bias as a feature for factuality prediction, not only is factually prediction hard in general but.

249

00:36:15.870 —> 00:36:24.690

Jack Cannon: Just because someone's biased politically doesn't mean that they're giving you wrong information, so I don't really like that they use that as a signal.

250

00:36:25.830 —> 00:36:29.730

Jack Cannon: book and that's because biases isn't necessarily bad for that reason.

251

00:36:30.660 —> 00:36:37.650

Jack Cannon: But bias is bad when you have a news media that says they're balanced but they're really not balanced that's that's when I think bias is a problem.

252

00:36:38.220 —> 00:36:49.950

Jack Cannon: And they're basically blind to the public that point because it's not like we have the time to be researching all these super complicated topics that's kind of what journalists are supposed to do so if they're not doing it right from that's wrong.

253

00:36:53.310 —> 00:37:03.210

Jack Cannon: They kind of made an assumption at the beginning of the paper they thought that the trust crisis was caused by social media and all the you know information that's been flooded into the public sphere.

254

00:37:03.510 —> 00:37:14.370

Jack Cannon: Such that you know traditional media can't protect us from it anymore, and that that's the trust crisis, and I think that the the traditional media is part of the problem themselves so you can see.

255
00:37:15.900 —> 00:37:19.710
Jack Cannon: You know this came out last week just a study.

256
00:37:20.220 —> 00:37:32.400
Jack Cannon: And there's a lot of studies that are similar to this to happen over time how people feel about traditional media, but they were kind of saying, trust is kind of an all time low, you know only 46% of people trust with the traditional media says.

257
00:37:34.170 —> 00:37:48.540
Jack Cannon: And furthermore fact checkers that are these fact checking websites no political fact they are just journalists themselves so they are, they are the traditional media so it's kind of like the circular circular thing, where you have.

258
00:37:49.110 —> 00:38:01.980
Jack Cannon: problematic traditional media, then you have them trying to regulate themselves so if they're biased, then that bias is going to recycle itself in these quote unquote facts that are being checked.

259

00:38:04.020 —> 00:38:06.780

Jack Cannon: over here on the left hand side or on the right hand side.

260

00:38:07.380 —> 00:38:19.140

Jack Cannon: I took this order I took this graph from Article that came out last year and this guy basically went and he did a word search on this lexis nexis database and this just has pretty much every news article.

261

00:38:19.500 —> 00:38:29.520

Jack Cannon: They they have been collecting every news article since the early 1900s so you can do keyword searches, so he did a keyword search on like.

262

00:38:30.330 —> 00:38:38.040

Jack Cannon: a bunch like 70 polarizing terms, and you can see, the trend and how many times, this shows up in a lot of these news articles.

263

00:38:38.340 —> 00:38:55.260

Jack Cannon: And it's like right around here like 2009 2010 you start to see this spike and all the charts look like this with these polarizing terms they all look like this, where there's the spike in

2010 and you have to ask yourself well this seems to confirm the theory that.

264

00:38:56.970 —> 00:39:08.040

Jack Cannon: Part of this explosion social media took audiences away from these traditional media of everybody who used to just watch CNN all the time or read the Washington Post every day.

265

00:39:08.310 —> 00:39:15.240

Jack Cannon: Now they don't really have that have to do that anymore, because they have other sources of information so in order to get these people back.

266

00:39:15.750 —> 00:39:26.730

Jack Cannon: You kind of have to resort to extreme measures and that's kind of what the theory is about why this is happening I block this out just because it's a polarizing term and i'm not trying to do that, to the class.

267

00:39:28.470 —> 00:39:40.620

Jack Cannon: So I guess what i'll say, the last thing i'll say is a good fact checking system should remove the human component completely so if someone could somehow design a system that is able to query a database of.

268

00:39:42.330 —> 00:39:52.380

Jack Cannon: source database, such as economic data if a system can retrieve that data automatically That would be good, but just looking at what in fact checker say is really good.

269

00:39:53.970 —> 00:40:03.780

Jack Cannon: And then, this also presents kind of ethical issue, so this made me cringe this came across my feed on Sunday, and it was the glue alphabet seo.

270

00:40:04.440 —> 00:40:10.770

Jack Cannon: You know, fighting information is cool at the heart of everything, Google does and in the article, they were talking about how.

271

00:40:11.340 —> 00:40:23.250

Jack Cannon: it's a major part of what their goal is, as far as what the search algorithm does, and of course YouTube is part of Google, so that is kind of baked in all this, and he said right now, they have.

272

00:40:23.670 —> 00:40:31.530

Jack Cannon: A large human component to do this, but they also have a large Ai component to doing that so systems like this are going to be used for.

273

00:40:32.640 —> 00:40:40.500

Jack Cannon: flagging things that are misinformation, if you will, so it 's an ethical issue because you kind of got to ask yourself the question.

274

00:40:42.060 —> 00:40:43.890

Jack Cannon: First off should something be.

275

00:40:45.450 —> 00:40:53.700

Jack Cannon: Trying to label, something is misinformation and what are they going to do when they and then, if you can do that, why would you want to do that.

276

00:40:54.000 —> 00:41:00.120

Jack Cannon: And that kind of beds That begs the question of you know, is something misinformation or as an information that you just don't like.

277

00:41:01.020 —> 00:41:08.490

Jack Cannon: Or is something hate speech, or is it just speak the speech that you hate, so this kind of can lead to censorship.

278

00:41:08.940 —> 00:41:22.890
Jack Cannon: And that's something we definitely want to be careful of when we're when we're looking at these types of things, so thank you very much for your time i'll leave you with a nice quote with from a pillar of classical liberalism and thank you for your time and do you have any questions.

279
00:41:46.260 —> 00:41:47.730
JoAnna Langberg: that's what I would say at the end of.

280
00:41:52.920 —> 00:41:54.600
JoAnna Langberg: It was hot.

281
00:41:58.440 —> 00:41:59.010
Jack Cannon: All right.

282
00:42:01.980 —> 00:42:03.330
Jack Cannon: lot of material.

283
00:42:04.860 —> 00:42:06.210
Jack Cannon: For this next data set.

284
00:42:09.420 —> 00:42:10.890

Jack Cannon: So i'm going to stop sharing.

285

00:42:13.530 —> 00:42:14.400

JoAnna Langberg: or stop recording.

286

00:42:14.970 —> 00:42:17.610

Jack Cannon: Start recording it was just testing you.