Automated Coding of Political Campaign Advertisement Videos: An Empirical Validation Study

June Hwang* Kosuke Imai† Alex Tarr‡

March 31, 2019

Abstract

Television advertisements play an essential role in modern political campaigns with several billion dollars spent in the 2018 general election alone. For over two decades, researchers have studied TV ads by analyzing the hand-coded data from the Wisconsin Advertising Project (WAP) and its successor, the Wesleyan Media Project (WMP). Unfortunately, manually coding more than a hundred of variables, such as issue mentions, opponent appearance, and negativity, for many videos is a laborious and expensive process. We propose to automatically code campaign advertisement videos. Applying state-of-the-art machine learning methods, we extract various audio and image features from each video file. We show that our machine coding is at least as accurate as human coding for many variables of the WAP/WMP data sets. Since many candidates make their advertisement videos available on the Internet, automated coding can dramatically improve the efficiency and scope of campaign advertisement research.

Key Words: audio data, computer vision, image data, machine learning, text data, video data

*Graduate Student, Department of Government, Harvard University, Cambridge MA 02138. Phone: 609–751–7028, Email: hwang@g.harvard.edu

†Professor, Department of Government and Department of Statistics, Harvard University, Cambridge MA 02138. Phone: 617–384–6778, Email: Imai@Harvard.Edu, URL: https://imai.fas.harvard.edu

‡Graduate Student, Department of Electrical Engineering, Princeton University, Princeton NJ 08544. Phone: 978–270–8483, Email: atarr@princeton.edu

1 Introduction

Modern political campaigns rely on advertisements of various forms, including television and radio commercials, leaflets, and emails, to get their messages heard and mobilize voters. In particular, TV advertisements are one of the most popular platforms because they allow political campaigns to target a certain group of voters in a specific media market or simply reach the large proportion of the voting-age population across the nation through a single medium.[1] According to one source, during the 2018 election cycle, a total of 8.5 billion dollars were spent on TV advertisements, making them the most prevailing medium to deliver campaign messages.[2] Given their importance in electoral campaigns, many researchers have studied various aspects of TV advertisements.[3]

Although there are several studies that utilized experimental setups (e.g., Brader, 2005; Clinton and Lapinski, 2004; Gerber *et al.*, 2011), a great number of researchers conducting observational studies (e.g., Banda, 2015; Kaplan *et al.*, 2006; Kang *et al.*, 2017; Krupnikov, 2011; Meirick *et al.*, 2018; Schaffner, 2005; Sides and Karch, 2008) rely on the comprehensive data on TV advertisements provided by the Wesleyan Media Project (WMP),[4] which succeeded the Wisconsin Advertising Project (WAP).[5] The WAP/WMP data cover all federal and gubernatorial elections and some recent state-level elections that span from 1998 to 2014. The data sets include a number of variables related to the contents of ads such as issue mentions, opponent appearance, mood of background music, and negativity. In addition, the data sets also contain information about the airing of each advertisement, such as broadcast time and frequency, the media market in which it was aired, and the name of the TV program that was running at the time of its airing.

The contents-related variables are manually coded by a group of research assistants who watch

[1] Derek Willis, "Why Television Is Still King for Campaign Spending," *New York Times*, 30 June, 2015.

[2] According to the same source, although online advertisements have been becoming popular, the estimated spending amount was 1.8 billion dollars (Borrell Associates, 2017).

[3] Examples include their mobilizing and persuasive effects (e.g., Ashworth and Clinton, 2006; Gerber *et al.*, 2011; Huber and Arceneaux, 2007),the impacts of their negativity (e.g., Fridkin and Kenney, 2011; Goldstein and Freedman, 2002; Krupnikov, 2011), and their issue dimensions (e.g., Banda, 2015; Kaplan *et al.*, 2006; Spiliotes and Vavreck, 2002).

[4] See `http://mediaproject.wesleyan.edu/` (accessed March 23, 2019)

[5] See `http://elections.wisc.edu/wisconsin-advertising-project/` (accessed March 23, 2019)

a large number of video files and record their coding results through a web-based platform. The source video files are provided by the Campaign Media Analysis Group (CMAG) of Kantar Media, a private corporation that specializes in the collection and analysis of TV ads airings. The WMP currently lists 47 undergraduate and graduate research assistants who participate in the coding of the advertisement videos.[6] The WAP/WMP data appear to be of high quality. For the 2014 election cycle, for example, the average intercoder agreement, based on a sample of 1,319 double-coded videos for 144 variables, is 95.8% (Wesleyan Media Project, 2017).[7]

While the WAP/WMP data have made important contributions to research on TV advertisements, manually coding more than a hundred variables for each of the video files is a laborious and expensive process, in which a team of research assistants watch thousands of videos and collect necessary information on the contents of each video. In addition, partly due to the contractual obligation, the data for each election are generally not available until the conclusion of the next election, leading to a delay of two to four years before other scholars can gain access to them.

To address these shortcomings, we propose to automate the coding of political campaign advertisements. Applying state-of-the-art machine learning methods, we extract various audio and image features from each video file and code them to reproduce the original variables in the WMP data. By directly comparing the results of our automated coding with the WMP human coding, we show that machine coding is at least as accurate as human coding for such variables as issue and opponent mention, face recognition, and negativity. The only variable for which human coding is significantly more accurate than machine coding is the mood classification of background music. However, this variable has a considerable degree of disagreement even among human coders.

We contribute to the small but rapidly growing political science literature on the analysis of audio, image, and video data (e.g., Dietrich, 2018; Dietrich *et al.*, 2018; Knox and Lucas, 2018; Torres, 2018) by conducting an empirical validation study for machine classification of video

[6] `http://mediaproject.wesleyan.edu/about/team-2/` (accessed on March 18, 2018)

[7] Similarly, based on a sample of 903 double-coded videos aired during the 2012 election cycle, the average intercoder agreement for 139 variables is 95.6% (Wesleyan Media Project, 2016b).

files. The rich human coding data from the WAP/WMP projects provide an unusual opportunity to directly compare automated coding with human coding using a large number of video files across a number of variables (see Proksch *et al.*, 2019, for a validation study of automatic speech recognition). Our findings suggest that coding tasks done by student research assistants can be achieved by machines to a similar degree of accuracy. We believe that social scientists should take advantage of this recent technological advancement in all aspects of research projects from data collection to analysis.

Overall Workflow. We briefly summarize the overall workflow for our validation study. Since the CMAG video files are of low resolution and are not suitable for automated coding, we first obtain high-resolution video files from the official YouTube channels of political campaigns. We use an audio matching algorithm to select a subset of video files that are high-resolution versions of the CMAG videos, which serve as our validation data set. Although the coverage is far from perfect, a majority of the CMAG videos that were aired for general elections and sponsored by the candidates or their parties are found on YouTube.

Once the validation data set is constructed, we extract audio and visual features from each video file. We combine image texts extracted from visual frames and textual data obtained through automatic audio transcription in order to identify what kinds of political, economic, and other important issues each video addressed. The same set of features are used to also detect whether the ad contains a reference to the opposing candidate. In addition, we use face recognition to detect appearances of the candidates and their opponents, as well as some well-known politicians such as Barack Obama. The specific types of music used in the ads for their respective purposes are analyzed through music mood classification based on the audio features. Lastly, we use both textual and music features to classify whether or not a video represents a negative advertisement. For each of these automated coding tasks, we evaluate its accuracy by directly comparing the results of machine coding with those of human coding, including the original variables provided by the WMP and in some cases, judgments made by a group of workers recruited through Amazon's

Mechanical Turk.

2 The Validation Data

In this section, we provide detailed descriptions of how we constructed the campaign advertisement video files used for our validation study. Our primary data source is the official YouTube channels of candidates, which provide high-resolution campaign ad videos free of charge. To create the validation data set, we match the downloaded video files against the comprehensive set of lower quality CMAG video files used by human coders for the WMP. The matching process identifies a set of videos that are high-resolution versions of the video files used by the WMP. Since the resulting data set corresponds to only a subset of the data used by the WMP, we analyze its coverage by identifying the characteristics of candidates and ads that predict the successful match.

2.1 Data Acquisition

The video data used for this paper come from two different sources. First, we obtained a set of campaign advertisement video files aired for Presidential, Congressional, and gubernatorial races during 2012 and 2014 election cycles recorded by the CMAG.[8] This set of video files, which are part of the data available from the WMP for a small fee, has a comprehensive coverage for all the television commercials aired in each of the designated media markets. Unfortunately, these CMAG videos have substantially downscaled image resolution and low audio quality, making them unsuitable for automated coding. The CMAG also does not make these videos publicly available and provide them only to the WMP under a special contract.

To address these issues, we use YouTube as an alternative source of data. When compared to the CMAG video files used by the WMP, the campaign commercial video files that are publicly available at YouTube are free of charge and tend to be of much higher quality. We first manually identify as many official YouTube channels of candidates as possible. These channels list various

[8]Following the WMP, for the states which hold their governor's elections in odd-numbered years, we include their campaign ad videos as part of the election cycle in the following year.

Election cycle	Office	All candidates	Candidates with YouTube channels		Video types 15-sec.	30-sec.	60-sec.	All videos
2012	President	2	2	(100%)	40	263	97	400
	House	317	263	(83.0%)	40	986	199	1225
	Senate	64	50	(78.1%)	16	520	147	683
	Governor	25	20	(80.0%)	15	143	36	194
2014	House	255	199	(78.0%)	59	813	175	1047
	Senate	68	52	(76.5%)	63	728	206	997
	Governor	86	59	(68.6%)	76	636	176	888
	Total	817	645	(79.0%)	309	4089	1036	5434

Table 1: Summary of Channels and Video Files Found at YouTube. The table presents the number of candidates for each office listed in the Wesleyan Media Project (WMP) data, the number and percentage of these candidates for whom we found YouTube channels, and the number of downloaded video files from the said candidates (different types based on their length). Note that not all of the downloaded videos are campaign TV advertisements.

videos uploaded by the candidates' campaigns. From each of these channels, we download all video files that are approximately 15, 30, or 60 second long, using ±5-second-buffers around the target to allow for small deviations from the standard lengths. These length restrictions based on the standard formats of TV commercial clips used by advertisers allow us to filter out many of the irrelevant videos, such as free-form online advertisements, clips of townhall meetings, and media coverage/interviews.

Table 1 summarizes the channels and video files found at YouTube using the procedure described above. We find that approximately 80% of the general election candidates listed in the WMP data have official YouTube channels. In addition, a greater proportion of House candidates have YouTube channels than the Senate and Gubernatorial candidates. Lastly, we found a total of several thousand video files that have approximately the same lengths as those of campaign TV advertisement videos. A majority of video files are 30 seconds long.[9]

[9]On average, we have found the greatest number of video files for the Senate races (15.6 files per candidate), followed by the Gubernatorial races (13.8 files), and the House races (8.1 files). Again on average, we have a slightly larger number of videos from Republicans (7.4 files per candidate) than from Democrats (6.7 files per candidate); from challengers or open-seat contenders (7.6 files per candidate) than from incumbents (6.6 files per candidate); and from non-winners (7.6 files per candidate) than from winners (6.8 files per candidate) of the elections they were running for.

2.2 Matching the YouTube Data to the CMAG Videos

To directly compare automated coding with the WMP coding, we further subset the retrieved YouTube videos by matching them to the CMAG videos for each candidate. The matching process filters out irrelevant videos that happen to fall within the length ranges we chose. Since a video file consists of both visual frames and audio encodings, matching one video to another in their entirety is a complicated process. This is particularly so in our case where one collection of videos (from YouTube) are of significantly higher resolution compared to the other collection (from the CMAG). In particular, given the substantial difference in image quality between these two sets of videos, we exclusively use the audio portions of the videos as the basis for declaring matches.[10]

Compared to exact matching with both visual and audio portions, audio matching is a simpler task, and there exist multiple algorithms that can match one file to another with high accuracy (Wang *et al.*, 2003), with Shazam being a notable example in commercial industry. We use the spectral fingerprinting algorithm of Haitsma and Kalker (2002) to perform matching between the YouTube and CMAG video collections, whose primary advantage over other alternatives is its ease of implementation. Spectral fingerprinting performs feature extraction by reducing a high-dimensional raw audio signal to a low-dimensional "fingerprint" with minimal loss in information pertaining to the identity of the audio track. Specifically, we compute the fingerprints from the spectrogram of the digital audio signal.[11] The spectrogram is a two-dimensional representation of the frequency content of the signal as it varies with time. This frequency representation of the signal provides important information as to the perceptual qualities (i.e., tempo, timbre, pitch, etc.) of the audio track.

We briefly describe the process through which the spectrogram is calculated. Figure 1 illustrates the entire process, and we begin with a brief introduction to audio signal processing. The

[10]There are some cases where two videos with almost identical audio portions have small differences in the visual components, such as added captions or the names of the different locations within the election district. Since campaign advertisements are designed to convey specific messages to voters in a short amount of time, however, such minor visual differences rarely change the main contents of messages.

[11]The spectrogram is computed with the Python package `scipy`.

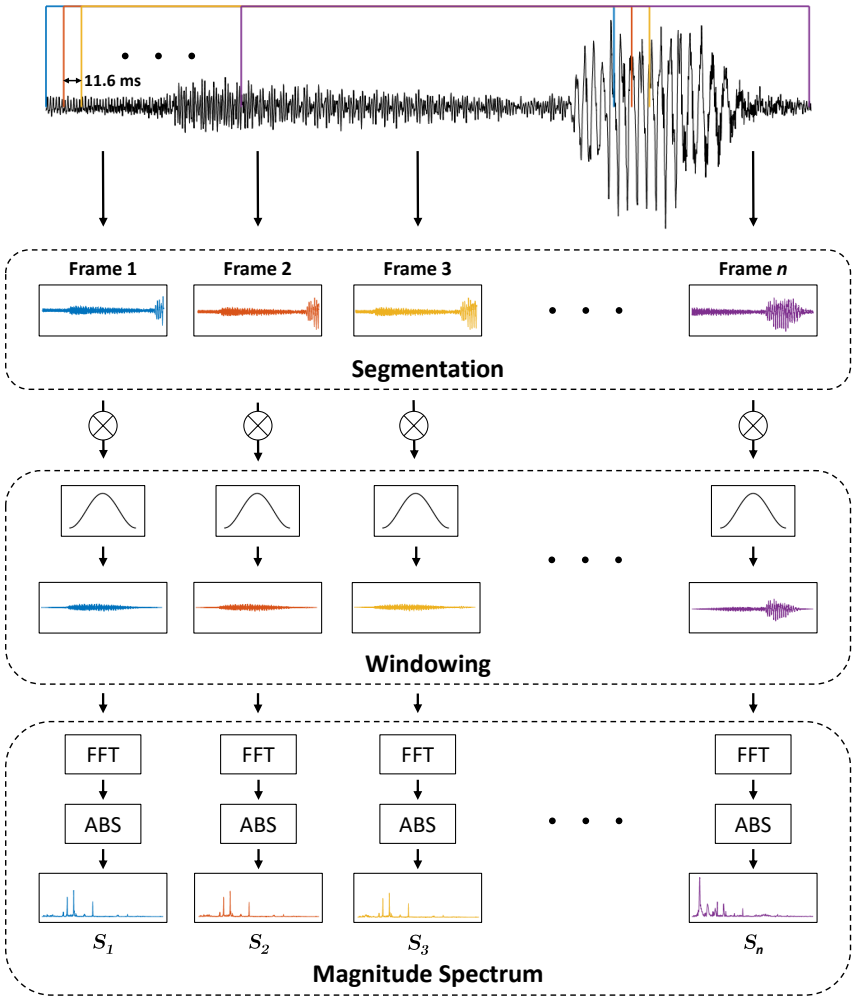


Figure 1: System Diagram for Computing the Spectrogram. The diagram illustrates the process of computing the spectrogram for a 0.5 second segment of audio for one of the campaign videos in our validation dataset. The signal is first partitioned into segments of length 0.3712 seconds, spaced apart by 11.6 milliseconds (ms), resulting in a collection of n frames, shown in the box labeled "Segmentation." Each segment is then multiplied by a Hann window to produce a smoothed waveform, as shown in the "Windowing" box. Finally, in the last box the magnitude spectrum is computed via a fast Fourier transform (FFT) followed by an absolute value (ABS). The magnitude spectrum for frame j corresponds to row j in the spectrogram and is denoted as S_j.

audio we hear in a video is a continuous waveform that propagates through the air as a compression wave, vibrating the air molecules around our ears, which creates the sound. These compression waves are generated from the vibration of speakers in our sound device, with the pattern for the speaker vibration dictated by a continuous electrical signal, called the analog signal. Since the

video files cannot store all the values of the continuous analog signal, a discretized approximation is used by sampling the desired continuous waveform at fixed intervals of time, where the number of samples obtained per second of audio is called the sampling rate. For the matching procedure, we used digital audio signals from the videos with a sampling rate of 5 kHz, implying that for every one second of audio, we obtain 5,000 equally-spaced samples. A more detailed introduction on digital signal processing can be found in Oppenheim *et al.* (1996).

Formally, given a digital audio signal $x[l]$ consisting of L samples, where L is determined by the duration of the audio file, the spectrogram is computed by first splitting the audio signal into overlapping segments, each of which has a length of $W = 1,856$ samples, corresponding to duration 0.3712 ($= W/5000$) seconds. We use short-duration segments because the frequency content of audio signals varies quickly over time. For example, music typically consists of several different instruments playing a sequence of notes in rapid succession. Splitting the full signal into smaller segments allows us to better isolate the individual instruments and notes, allowing us to better characterize the short-term variations of the audio signal.

We use an overlap factor of 31/32, meaning that consecutive segments share 31/32 of the same samples. Hence, the temporal spacing between the first sample of each consecutive segment is 11.6ms = 0.3712/32. The reason we use a large overlap factor is that it helps with the matching procedure, as we explain later. Each frame is then multiplied by a Hann window to reduce high frequency noise introduced due to the segmentation (Harris, 1978). Finally, we compute the one-sided magnitude spectrum for each frame by taking the absolute value of the 2048-point fast Fourier transform (FFT) of the input frame. This yields a spectrogram $S \in \mathbb{R}^{N \times K}$, with element $S(n, k)$ corresponding to the magnitude of frequency component k in frame n.[12]

To reduce the dimensionality of frequency, we partition the resulting spectrogram into $M = 33$ non-overlapping, logarithmically-spaced frequency bands covering the frequency range 300Hz to 2,000Hz. Within each band m, we compute the energy, defined as $E(n, m) = \sum_{k=1}^{K_m} S(n, k)^2$, where

[12]The number of frequency bins K is fixed to 1025 for all fingerprints, while the time length N varies with the duration of the audio signal.

K_m is the number of frequency bins. This step yields a matrix of energy values for each frequency band and spectrogram time segment. We then produce a binarized matrix $B \in \mathbb{R}^{N-1 \times M-1}$ through the following rule,

$$B(n,m) \;=\; \begin{cases} 1 & \text{if } E(n+1,m) - E(n+1,m+1) - (E(n,m) - E(n,m+1)) > 0 \\ 0 & \text{if } E(n+1,m) - E(n+1,m+1) - (E(n,m) - E(n,m+1)) \leq 0 \end{cases}, \quad (1)$$

which indicates the sign of energy differences between adjacent bands and frames. Since there are 33 frequency bands, this procedure results in a 32-element binary array for each $n \in \{1, \ldots, N-1\}$. We convert these binary arrays into 32-bit unsigned integers and call the resulting vector F the fingerprint, with each element of the vector called the sub-fingerprint.

For each YouTube video, we compute the spectral fingerprints for the full-duration of all of the downloaded YouTube videos, producing a lookup table (LUT) of fingerprints to be used to match the CMAG videos to. The hash values in the LUT represent the sub-fingerprints while the values stored are the ID of the file and the index in the full fingerprint where the sub-fingerprint occurs. Note that because the videos have varying lengths, the spectral fingerprints will have varying lengths as well. For each unmatched CMAG video, we then compute the spectral fingerprint of the middle 12-second segment of the unmatched CMAG audio track and attempt to find its matching fingerprint in the YouTube video database.[13]

Since the sub-fingerprints are computed at discrete intervals of time, there is no guarantee that segments obtained in the fingerprinting process of the truncated audio correspond to the same segments in its matching file in the YouTube database. We see now that the purpose

[13]There are several reasons why the duration input audio signal to the spectral fingerprinting process is fixed to 12 seconds. First, the entire clip is not needed to find a match, so using a shorter segment leads to significant computational performance. Second, many of the CMAG videos were improperly recorded and often contain extra silence or portions of other ads at the boundaries of file (the end or the beginning), so using the entire audio file would lead to false negatives in the matching process because, technically, the audio files contained different content. Third, while shorter duration could be used to effectively match videos, we found that some of the ads in the CMAG dataset had some segments that were identical to one another, while other segments were different. By using longer duration clips for matching, we reduce the probability of false positives.

of the large overlap factor is mitigate issues with spectrogram frame boundary misalignment between the CMAG video and the YouTube database video in the fingerprinting process. Since the spacing between frames is so small, the sub-fingerprints for the unmatched video should be very similar to the sub-fingerprints of its corresponding match in the database, even under worst-case misalignment.

The matching itself was performed in several steps (see Appendix A.1 for details). We first generate a list of candidate matches based on the sub-fingerprints of the unmatched fingerprint. Then, for each candidate, we compute the bit error rate (BER) between the unmatched fingerprint and the candidate fingerprint, which equals the average number of binary digits that differ between the two fingerprints. Following Haitsma and Kalker (2002), we declare the candidate fingerprint to be a match to the CMAG video if the BER was below a threshold of 0.3. If the candidate was not a match, we repeat this process for the next candidate until a match is found or until all candidates have been exhausted.

We empirically evaluate the accuracy of the matching procedure by randomly selecting 100 matching results for the 2012 Congressional and gubernatorial elections; 50 of them from videos with matches found and the remaining 50 from videos with no matches found, and then manually verifying whether the classifications are correct. We find that all 50 classifications are correct, indicating the success of our matching procedure.

Although our matching procedure yields accurate results, one issue is the presence of multiple versions of the essentially identical video files in the CMAG data set. For example, when a candidate modifies a single image text, the CMAG data set treats this as a new video file. This coding practice is not ideal for our purposes because we wish to validate our automated coding method using different videos. To address this issue, we match CMAG video files against themselves and identify multiple versions of the same video files. Across 2012 and 2014 election cycles, we find 344 pairs of CMAG videos declared as matches and consolidated them as single video files.

Election cycle	Office	All Candidates CMAG videos	All Candidates Matches found		Republicans CMAG videos	Republicans Matches found		Democrats CMAG videos	Democrats Matches found	
2012	President	228	184	(80.7%)	98	70	(71.4%)	130	114	(87.7%)
	House	1106	605	(54.7%)	574	285	(49.7%)	506	319	(63.0%)
	Senate	586	322	(55.0%)	279	127	(45.5%)	289	188	(65.1%)
	Governor	184	100	(54.4%)	94	46	(48.9%)	90	54	(60.0%)
2014	House	912	526	(57.7%)	437	252	(57.7%)	470	274	(58.3%)
	Senate	666	475	(71.3%)	327	230	(70.3%)	307	235	(76.5%)
	Governor	742	383	(51.6%)	383	188	(49.1%)	317	188	(59.3%)
	Total	4424	2595	(58.7%)	2192	1198	(54.7%)	2109	1372	(65.1%)

Table 2: Coverage of Political Advertisement Videos Available at YouTube. The table presents the number and percentage of CMAG videos in the WMP data set, for which we found identical but higher-resolution versions at YouTube. Among the three categories of columns, the first represents the numbers for all candidates, the second for Republican candidates, and the last for Democratic candidates. Note that the first category also includes third-party or independent candidates.

2.3 Coverage Analysis

A chief concern about using YouTube as the data source is its potentially insufficient coverage. Although a majority of politicians have YouTube channels (see Table 1), those who do not have YouTube channels may systematically differ from those who have them. In addition, politicians may not upload all of their campaign advertisement videos to their YouTube channels, and as a result, the coverage rate may vary across politicians. We examine how this sample selection is associated with the candidate characteristics.

Although we found official YouTube channels for about 80% of all general election candidates in the WMP data set, after removing the YouTube video files we were unable to match with any CMAG video, the proportion of the candidates who have at least one matched video file is reduced to approximately 66%. The coverage rate has improved from the 2012 elections to the 2014 elections for each office type by 2 (Senate) to 12 (Governor) percentage points despite the fact that the latter was a midterm election. This may suggest that the coverage may continue to improve over time as more political campaigns start using YouTube as a way to reach voters.

Table 2 presents the number and percentage of CMAG videos in the WMP data set, for which

we found identical but higher-resolution versions at the candidates' official YouTube channels. In the 2012 general elections, the coverage rate is the highest for the videos by the Presidential candidates while the coverage rate hovers around 55% for the other candidates. In the 2014 election cycle, we were able to find more than 70% of videos for Senate candidates whereas the coverage rate stays similar to that of the 2012 elections for the House and Gubernatorial elections. For all elections reported, we retrieved higher proportions of videos sponsored by Democratic candidates than those run by Republican candidates; on average, we have about 55% of all Republican TV ads recovered, and 65% of those from Democrats.

Appendix A.2 presents the results of additional coverage analysis. The results of regression analysis suggest that being incumbent and major party candidates is associated with a greater likelihood of having an official YouTube channel whereas the difference between Democratic and Republican candidates is relatively small. For video matching, we are more likely to find Democratic candidates' advertisement videos on YouTube than the ones about Republican candidates. The incumbency status does not seem to matter much for video matching.

3 Feature Construction

To analyze video files, the first step is to construct relevant audio and visual features. In this section, we describe the methods we use for feature extraction and construction.

3.1 Visual Features

3.1.1 Video summarization

As previously mentioned, video data is comprised of both audio and visual components. While audio data comes in the form of a digital signal, as discussed in Section 2.2, visual data comes in the form of a sequence of images, called *frames*. Most of the videos in our YouTube dataset were created using a frame rate of either 24 or 30 frames per second (FPS) at a resolution of 1280×720 pixels. Since a standard campaign advertisement runs for 30 or 60 seconds, loading all of the

frames (i.e., 720 — 1,800 frames) into a computer would require several gigabytes of memory for just a single video. Processing and analyzing this amount of data would be both cumbersome and computationally expensive in terms of both time and resources, making video data a difficult source of information for applied researchers to work with. Additionally, the high frame rate implies consecutive frames are highly correlated with one another and contain roughly identical visual information. Therefore, we first obtain a smaller set of frames that captures most of the visual information contained in the video through the procedure called *video summarization*, to which we now turn.

The algorithm we use to select the subset of frames representative of a video comes from Chakraborty *et al.* (2015). While there are many other alternative video summarization algorithms (Baskurt and Samet, 2019), we chose this algorithm due to its simplicity, ease of tuning, and flexibility in controlling the number of frames in the summary. This last part is particularly important for our application because if some frames are missed in the summarization process, then visual information is lost and can lead to misclassification errors in our later analyses. By using an algorithm that allows us to sacrifice summary uniqueness at the benefit of reducing the chance of missing frames containing crucial information, we can help mitigate one source of misclassification error in our automated coding tasks.

We briefly summarize the algorithm used for video summarization. For any given video consisting of a set of N frames, denoted by V, the problem of video summarization can be formulated as an optimization task. The optimal summary maximize uniqueness and representativeness while keeping the number of frames in the summary at a reasonable level. Here, the uniqueness quantifies how distinct frames are from one another in the summary, while the representativeness enforces the rule that for each frame in the video, there should exist at least one frame in the summary that is visually similar. The component for summary length is used to regularize the objective function since uniqueness and representativeness are maximized by selecting all frames in the video.

Formally, the video summarization task is formulated as the following optimization problem,

$$S^* \;=\; \underset{S \subseteq V}{\operatorname{argmax}} \underbrace{\sum_{i \in V} \max_{j \in S} w_{ij}}_{\text{representativeness}} + \lambda_1 \underbrace{\sum_{i \in S} \min_{j \in S} d_{ij}}_{\text{uniqueness}} + \lambda_2 \underbrace{(N - N_S)}_{\substack{\#\text{ of unselected} \\ \text{frames}}}, \tag{2}$$

where w_{ij} is a measure of representativeness between frames i and j, d_{ij} is a measure of uniqueness between two frames, N_S is the number of frames selected for summary S, and (λ_1, λ_2) control the relative weighting of the different terms.

In order to compute the representativeness w_{ij} and the uniqueness d_{ij} between two frames i and j, we first compute a feature representation for each frame. For the representativeness w_{ij}, we use a feature descriptor commonly used in computer vision for object detection called the histogram of oriented gradients (HOG) (Dalal and Triggs, 2005), which encodes the distribution of gradient directions for local color intensity. We compute pairwise distances between all frames in the video using the cosine similarity measure. For the uniqueness d_{ij}, we use the *Lab* histogram. *Lab* is a three-component color space that is more perceptually uniform with respect to human color vision than the standard RGB representation of images. We compute a three-dimensional histogram using 23 bins along each color dimension (L, a, and b) for every frame of a video file. To measure the distance d_{ij} between the *Lab* histogram representations of frames i and j, we use the χ^2-distance,[14], i.e., $d_{ij} = \sum_{b=1}^{23^3} (h_i(b) - h_j(b))^2 / (h_i(b) + h_j(b))$ where $h_i(b)$ and $h_j(b)$ are the counts in bin b for frames i and j, respectively.

Since the problem of finding the optimal subset that maximizes the objective function in equation (2) is known to be NP-hard, we use an approximation algorithm proposed by Chakraborty *et al.* (2015). The algorithm works by maintaining two solution sets initialized to $S_0 = \emptyset$ and $S_1 = V$. At each iteration, it randomly selects a frame without replacement from V and proposes either adding the frame to S_0 or removing it from S_1 with complementary probabilities based the relative change in the objective function that would result from the operation. After all frames

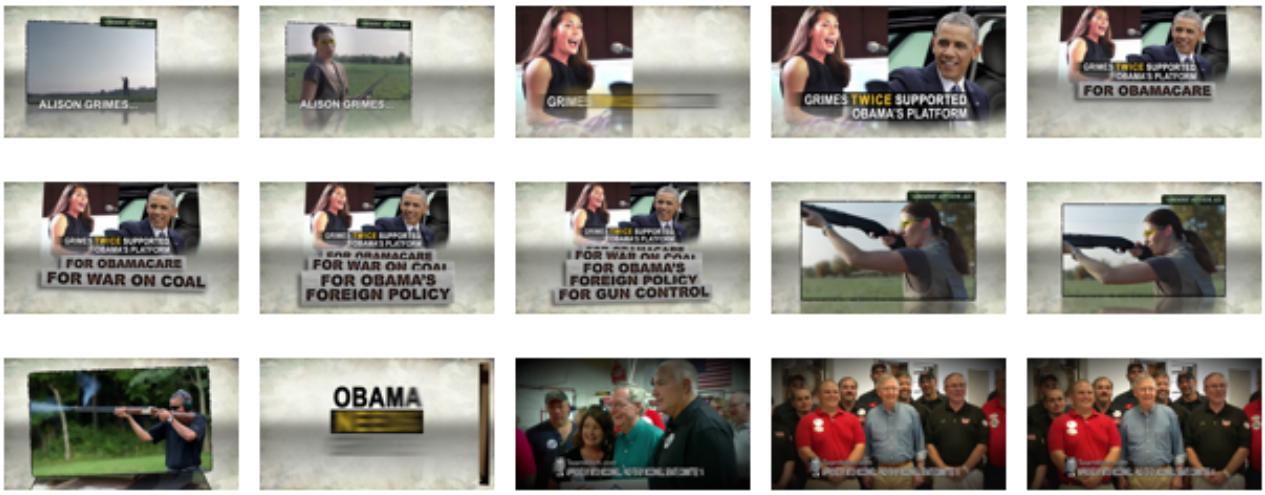
[14] Also called the additive χ^2 kernel.

are removed from V, the sets S_0 and S_1 will coincide, yielding a video summary. While the algorithm does not necessarily return the optimal summary, the flexibility in choosing the tuning parameters lets us to control the coarseness of the resulting summary. Following the suggestion given in Chakraborty *et al.* (2015), we set the tuning parameters to be, $\lambda_1 = 1$ and $\lambda_2 = 5$.

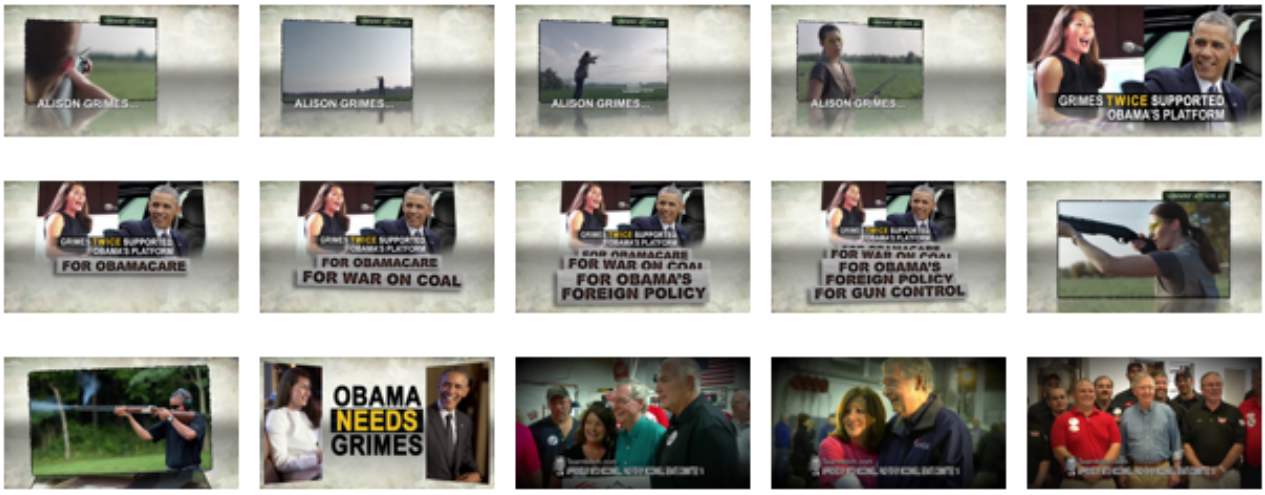
To evaluate the performance of our video summarization procedure, we compare the results of our auto-generated video summarization against those of manual summarization for a random sample of 20 video files from our validation data set. We watch each sampled video carefully and select frames that we think are representative of the visual contents. Our selection criteria is based on the idea that since a video consists of a series of shots, each of which represents uninterrupted segments of frames featuring a fixed camera angle that run between two edits or cuts, it can be effectively summarized by a single frame. Therefore, we construct the manual summary by selecting a single frame for each shot, where the number of shots varies across videos.

Figure 2 presents an example, which corresponds to the worst performance by automated video summarization among our 20 sampled videos. The original video file is taken from a campaign ad for Mitch McConnell in the 2014 Senatorial election in Kentucky. We find that in this case the auto-generated summary is missing several images in the beginning showing a woman aiming a gun. However, this omission is not consequential because there are several other images later in the summary showing the same content. Additionally, the auto-generated summary is missing an image near the end of the video featuring the candidate posed with his wife and the approval message overlaid on the image. This omission can be explained by the fact that the auto-generated summary includes another image that contains essentially identical contents. For the other 19 comparisons, the performance of automated video summarization is even better, capturing most of the visual information contained in each video file.

The results of our validation study are summarized in Figure 3. Figure 3a shows a comparison of the number of frames selected between the auto- and manually-generated summaries. We see that for all videos, the summarization algorithm selected the same or more number of frame as the



(a) Auto-generated summary



(b) Manually-generated summary

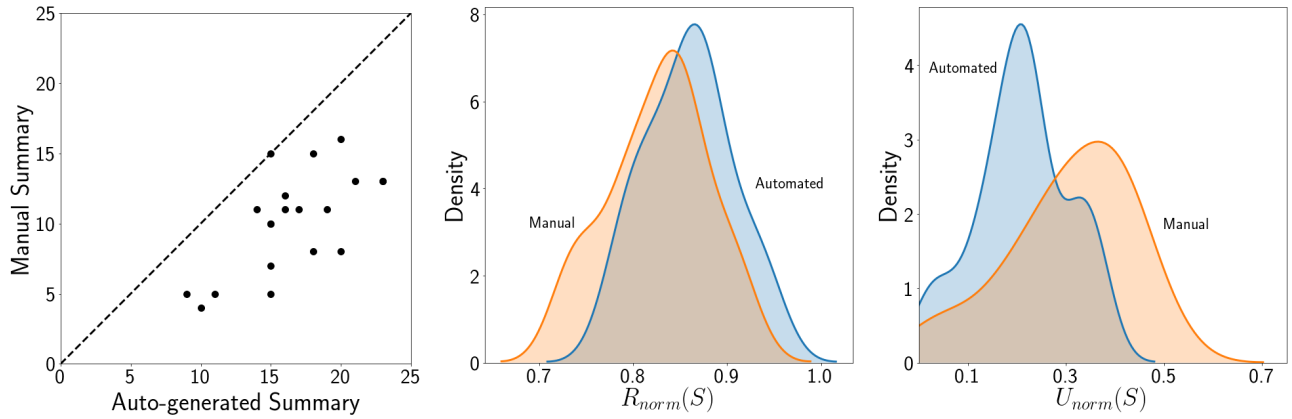
Figure 2: An Example of Comparison between Auto-generated Video Summary and Manually-generated Summary. Although this example corresponds to the worst performance by the automated video summarization method, it still captures much of the visual contents contained in the manually-generated summary.

manual summary. Since the original video files contain roughly 720–960 frames,[15] both summaries resulted in a roughly 97%–99% reduction in the number of frames.

In Figure 3b we plot the distribution of the normalized summary representativeness, i.e., $R_{norm}(S) = R(S)/N_{total}$ where N_{total} is the total number of frames in the video.[16] We find that the

[15] All videos in this sample were captured at 24 or 30 FPS and had a duration of ∼30 seconds.

[16] This normalization is used since $N_{total} = \max_{S \subseteq 2^V} R(S)$, so $R_{norm}(S)$ is a relative measure of how represen-



(a) Number of summary frames (b) Representativeness (c) Uniqueness

Figure 3: Results of the Validation Study for the Video Summarization Algorithm. Plot (a) shows the comparison between the number of frames selected into the summary for the auto- and manually-generated summaries. Plots (b) and (c) show the comparison between the distributions of representativeness and uniqueness over automated (blue) and manual (orange) summaries, respectively.

auto-generated summaries tend to be slightly more representative than the manually-generated summaries, suggesting the algorithm does well in representing the content of the video.

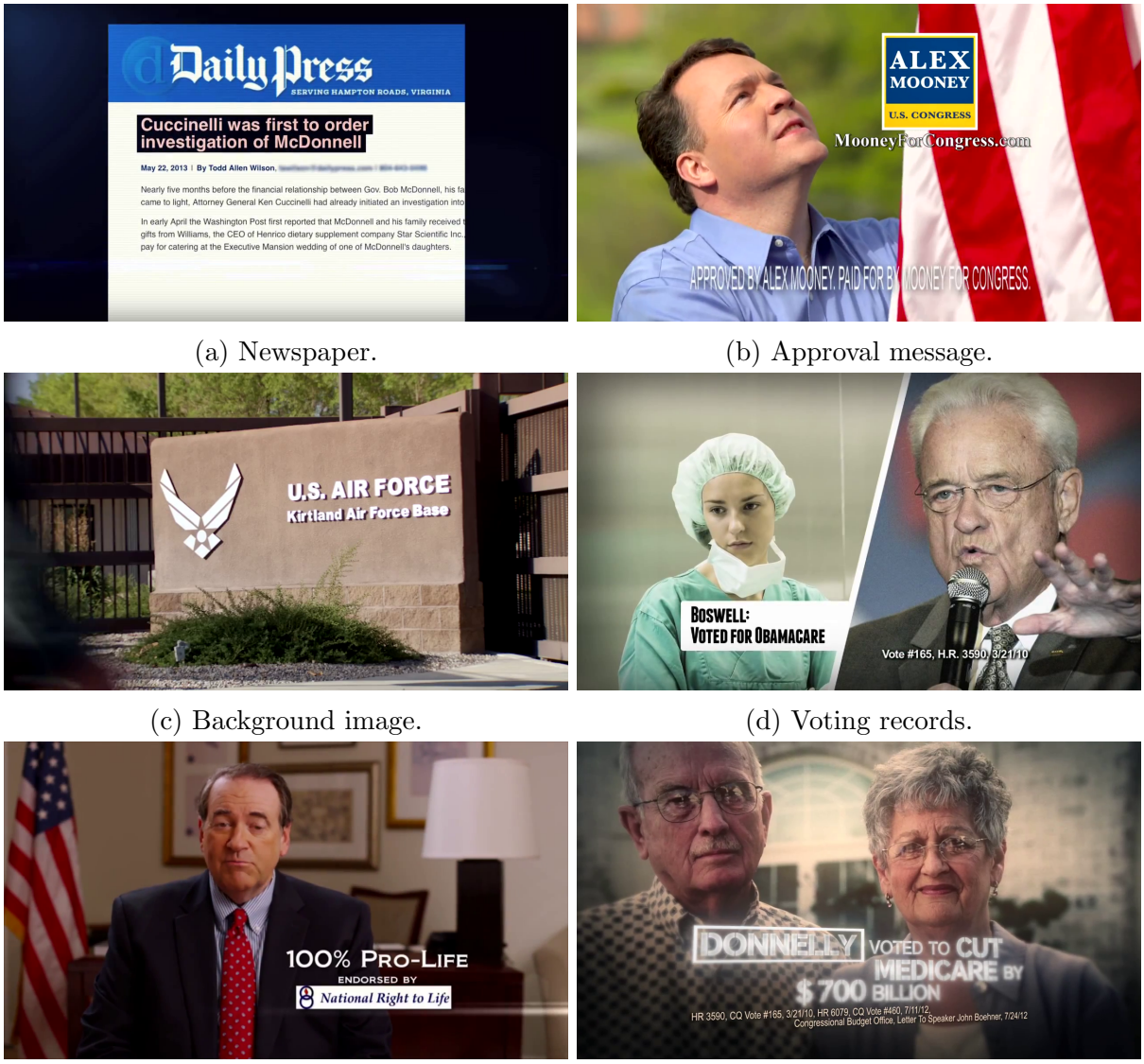
Figure 3c shows the distribution of the normalized uniqueness measure, i.e., $U_{norm}(S) = R(S)/N_S$, where N_S is the number of frames in the summary. A high value of $U_{norm}(S)$ suggests there are few duplicates in the summary. We find that the auto-generated summaries tend to be less unique than the manually-generated summaries. This is not surprising since the algorithm tended to produce a greater number of frames than manual summarization.

Finally, visual inspection of auto-generated and manual summaries indicates that the algorithm produced summaries that contain at least one image for each shot in video in 80% of the sample. In all instances where an image is missing, the missing image either contains no meaningful information pertaining to the content of the video or there is another image in summary that carries the same visual information.

3.1.2 Image text detection

Many campaign advertisements use image texts or on-screen texts to provide additional information, emphasize certain contents of ads, and display approval messages and endorsements. Since

tative the summary is compared to the entire video.



(a) Newspaper. (b) Approval message.

(c) Background image. (d) Voting records.

(e) Endorsement. (f) Policy position.

Figure 4: Examples of Image Texts. Source: see footnote 17.

our goal is to analyze the contents of ads, it is important to extract image texts and use them as features that supplement audio texts for our classification tasks. Figure 4 presents some examples of frames containing image texts, ranging from the use of newspaper to the display of policy position and endorsement.[17]

[17]Figure 4a: Republican candidate Ken Cuccinelli in the 2014 gubernatorial election in Virginia. Figure 4b: Republican candidate Alex Mooney in the 2014 House election for the 2nd district in West Virginia. Figure 4c: Democratic candidate Martin Heinrich in the 2012 senatorial election in New Mexico. Figure 4d: Republican candidate Tom Latham in the 2012 House election for the 3rd district in Iowa. Figure 4e: Republican candidate Chuck Fleischmann in the 2014 House election for the 3rd district in Tennessee. Figure 4f: Republican candidate Richard Mourdock in the 2012 senatorial election in Indiana.

We use the Google Cloud Platform (GCP) Vision API[18] to perform image text detection on each frame in a given video summary and obtain the raw text data. The GCP is a cloud-computing service that offers a wide variety of data processing tasks for a small fee. Because the GCP is a proprietary service, little to no information is available about the algorithms they use. However, many believe that the Vision AP algorithms are based on convolutional neural networks, which is the standard approach in the current literature on image text detection and recognition (Ye and Doermann, 2015; Zhu *et al.*, 2016).

We illustrate the performance of the API using the examples shown in Figure 4. While the performance of the algorithm is not perfect, it captures most of image texts. Specifically, the API is able to recover all texts in the frames of Figures 4a, 4c, and 4e, while missing just a few words in the frames of Figures 4b, 4d, and 4f. In some cases, the API is too accurate. The algorithm detected all of the small document texts in the newspaper shown in the frame of Figure 4a, which human coders would not be able to process in a short amount of time. The algorithm tends to miss texts which blends into the background. For example, the API fails to detect "Approved by Alex Mooney. Paid for by Mooney for Congress" in the frame of Figure 4b, which uses narrow fonts and mixes with different backgrounds. The algorithm also misses the date, "3/21/10," in the frame of Figure 4d while correctly detecting all the other words. Finally, the API misses the phrase "Donnelly voted to cut" in the frame of Figure 4f perhaps because of the box used around the "Donnelly" and its fuzzy fonts. Fortunately, in this and many other cases, the missed phrase was also spoken by people in the video.

Finally, we note that poor video summarization can impact the performance of image text detection. If the image quality of extracted frames is poor, any algorithm will have a difficult time detecting image texts. A good illustration of this is shown in Figure 2. While the manually-generated summary contains the frame with the image text of "OBAMA NEEDS GRIMES," the auto-generated summary included the frame, in which only the word "Obama" can be clearly

[18]Available at `https://cloud.google.com/vision/` (accessed March 22, 2019)

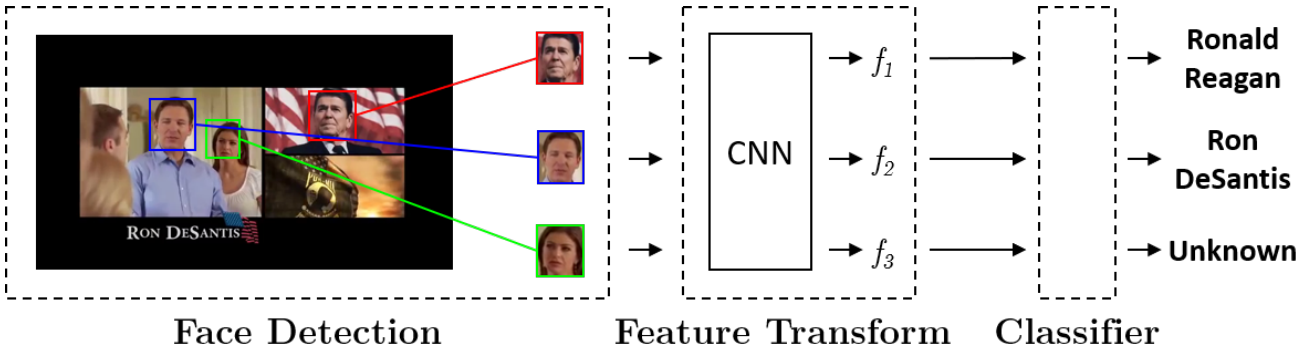


Figure 5: System Diagram for Face Recognition. An example is taken from a campaign video for Republican candidate Ron DeSantis for the 2012 House election in Florida's 6th district. First, faces are detected in the source image, producing a set of cropped images. A convolutional neural network (CNN) then computes a vector representation f_j of the detected face j in the feature transform step. Finally, the identity of the face is determined using a pre-trained classifier.

shown. Indeed, video summarization is a critical step for any visual feature extraction.

3.1.3 Face recognition

In addition to extracting image texts from the selected frames in the video summary, we also perform face recognition to determine candidate and opponent appearances in the ad. This automates the coding of variables in the WAP/WMP, which indicate whether particular politicians are mentioned or pictured in ads. Detecting the presence of opposing candidates is of particular interest as this is usually an indicator of an attack ad. Our facial recognition procedure consists of several steps as illustrated in Figure 5. First, we detect faces in each summary frame, producing a set of cropped images. Second, we use a convolutional neural network (CNN) to compute a feature representation for each of the detected faces. Finally, we use these features as inputs to a pre-trained classifier to determine the identity of the detected face.

We now describe the process for detecting faces and computing the feature representation, leaving discussion on classification and its results to Section 4.3. Face detection and recognition is performed using the Python package `facenet`,[19] which implements the multi-task cascade of neural networks (MTCNN) algorithm for face detection (Zhang *et al.*, 2016) and the FaceNet algorithm for the feature computation (Schroff *et al.*, 2015), both which are convolutional neural

[19] Available from `https://github.com/davidsandberg/facenet` (accessed March 22, 2019)

network-based approaches. While there are many other face detection (see Jin and Tan, 2017) and face recognition (see Wang and Deng, 2018) algorithms, we opted to use MTCNN and FaceNet mainly due to the availability of open-source implementations with pre-trained models.

We summarize the two algorithms for face detection and face recognition. The MTCNN algorithm takes an image as input and produces a probability of a face appearing in the image and two sets of vectors: $\{b_i \in \mathbb{R}^4 \mid i = 1, 2, \ldots, n\}$ which represents parameters defining the bounding box of each of the n detected faces,[20] and $\{l_i \in \mathbb{R}^{10} \mid i = 1, 2, \ldots, n\}$ which represents pixel coordinates for detected facial landmarks.[21]

The image in Figure 5 shows the detected bounding boxes in red, blue, and green. Each network in the cascade aims to produce bounding boxes for faces in the image, with the first network generating many candidates, while later networks aim to refine and remove false positives. In between each network is a calibration step that rejects false candidates and merges bounding boxes with significant overlap. The training process for these networks leverages a weighted loss function that balances three tasks: binary classification of whether or not a face is contained in the image, bounding box prediction, and facial landmark prediction. Specifically, for a given labeled training dataset of N samples $\mathcal{D} = \{x_i, d_i, b_i, l_i\}_{i=1}^N$, where x_i is the image, d_i is a binary label indicating the presence of a face, b_i is a bounding box for the face, and l_i are the facial landmark locations, the total loss under network parameters β is given by

$$L_\beta(\mathcal{D}) \;=\; \sum_{i=1}^{N} -\Big\{ d_i \log \hat{d}_i + (1 - d_i)(1 - \log \hat{d}_i) \Big\} + \frac{\mathbf{1}\{d_i = 1\}}{2} \left(\|b_i - \hat{b}_i\|^2 + \|l_i - \hat{l}_i\|^2 \right),$$

where a hat indicates the value was predicted from the network. Using this loss function, the network parameters were learned using stochastic gradient descent with the WIDER FACE (Yang

[20] The elements of b_i correspond to row pixel of the top-left corner of the bounding box, the column pixel of the top-left corner, the height of the bounding box, and the width, respectively.

[21] The facial landmarks l_i correspond to locations of the two corners of the mouth, the two eyes, and the tip of the nose. Facial landmarks are used for realigning detected faces so that they are front-facing, however the `facenet` package does not use this feature, instead training the feature computation CNN with a dataset containing misaligned faces.

et al., 2016) and CelebA (Liu *et al.*, 2015) datasets as training data.

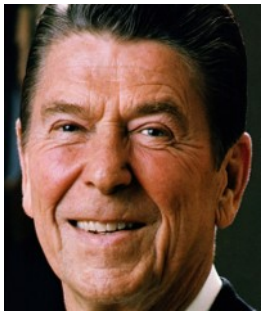
The FaceNet algorithm also uses a CNN. The goal of this network is to compute a vector representation of the given input image that is close in Euclidean distance to the vector representation of other images pertaining to the same person, and far from images of other people. This is accomplished using a CNN with the Google's Inception ResNet V1 architecture (Szegedy *et al.*, 2015) trained on the VGGFace2 dataset (Cao *et al.*, 2018), a massive and diverse collection of several million face images with variations in pose, emotional expression, illumination, and occlusion.

The network is trained to learn an embedding $f(x_i) \in \mathbb{R}^{128}$ where $\|f(x_i)\| = 1$ for a given face image x_i. The training process uses a *triplet loss* function, which, for a given dataset of face images and identities $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$, is defined as,

$$L_\beta(\mathcal{D}) \;=\; \sum_{j=1}^{N_{\text{trip}}} \max\left(0, ||f(x_j^a) - f(x_j^p)||^2 - ||f(x_j^a) - f(x_j^n)||^2 + \alpha\right),$$

where the sum is taken over all *triplet pairs* $\{(x_j^a, x_j^p, x_j^n)\}_{j=1}^{N_{\text{trip}}}$ of images. For a given image x_j^a from the training dataset, called the anchor image, a single triplet pair is formed by selecting another positive image x_j^p, which corresponds the same person shown in x_j^a, and a third negative image x_j^n, which corresponds to a different person. The parameter α corresponds to the margin between the distances and works to enforce a classification rule $\|f(x_j^a) - f(x_j^p)\|^2 + \alpha < \|f(x_j^a) - f(x_j^n)\|^2$ and is set to $\alpha = 0.2$.

Since the number of triplets formed from a dataset is large, with many triplets contributing little or nothing to the loss, only a subset of triplet pairs that are hard to classify are needed to update the network in each iteration of the algorithm. An example of a hard triplet pair is shown in Figure 6. Because Sam Brownback, who was the 2014 Republican candidate for the gubernatorial election in Kansas, has similar facial features to Ronald Reagan, the anchor-negative distance $\|f(x_i^a) - f(x_i^n)\|$ is relatively small, while the anchor-positive distance $\|f(x_i^a) - f(x_i^p)\|$ may be too large due to facial misalignment in the positive image. By using triplets such as these in each step



(a) Anchor image (x_j^a)

(b) Positive image (x_j^p)

(c) Negative image (x_j^n)

Figure 6: An Example of a Hard-to-classify Triplet Pair. The negative image (c) depicts Sam Brownback who was a republican candidate for the 2014 gubernatorial election in Kansas. The anchor image (a) is a picture of Ronald Reagan taken from Wikipedia, and the positive image (b) comes from a campaign ad for Democratic candidate Mary Burke in the 2014 gubernatorial election in Wisconsin.

of training, the network adjusts its parameters so that two Reagan pictures are closer together while Sam Brownback is further away, effectively learning how to distinguish between faces.

Using networks trained with these two algorithms, for each video summary we compute face embeddings for all faces detected across the summary, generating a facial feature set to be used for face identification. We note that like image text detection, face detection and recognition will also be negatively impacted by a poor quality summary. A good illustration of this is again shown in Figure 2. In the same frame that is missing the text "OBAMA NEEDS GRIMES", the pictures of Obama and Grimes are also missing although in this specific case they are pictured in other frames, and hence this will not affect our classification task.

3.2 Audio Features

Audio data contains critical information about the content of a campaign advertisement. A typical ad features a narrator, often the candidate, discussing their policy positions and highlighting issues with their opponent. Another prominent component is music. Nearly all campaign ads use music to set the tone of the ad. For example, ominous and tense music is used as a backdrop for an attack against the opponent. Below, we discuss the methods we use to extract transcripts from the audio and to compute features that capture the overall tone of the audio. We also discuss how text features are computed from transcripts.

"...it's about getting new jobs getting good jobs given middle class people the chance to get her kids a decent life nobody can tell me it's not a senator's job to create jobs and I choose Allison because she will work with people in both parties to do what's right for you since Alison to the Senate"

(a) Auto-generated transcript.

"...it's about getting new jobs getting good jobs giving middle class people the chance to give their kids a decent life nobody can tell me it's not a senator's job to create jobs and I choose Alison because she will work with people in both parties to do what's right for you send Alison to the Senate"

(b) Manually-generated transcript.

Figure 7: Excerpt of Auto- and Manually-generated Transcript. The auto-generated transcript was obtained by applying the Google Cloud Platform Video Intelligence API to a political ad for Democratic candidate Alison Lundergan Grimes in the 2014 senatorial election in Kentucky. The transcription errors are highlighted in red and the corrections appear in blue.

3.2.1 Speech transcription

We use the GCP Video Intelligence API to generate transcripts for each video in our dataset.[22] This algorithm takes a video as input and returns its estimate of the transcript for the video, without punctuation. Like the Vision API discussed in Section 3.1.2, an exact description on its algorithm for automatic speech recognition (ASR) is not available, though the literature suggests they are using a type of neural network called long short-term memory (LSTM) (Soltau *et al.*, 2016; Xiong *et al.*, 2016).

The GCP Video Intelligence API has a feature that allows the user to also provide collection of phrases expected to appear in the video to help improve the accuracy of the transcription. This is useful in our application since one of our goals is to detect candidate or opponent mentions in the ad. Since names are not commonly used words and some candidate names are derived from other languages, it can be difficult for ASR systems to accurately recognize them. For each video, we provide the names of both the candidate featured in the video and their opponent(s).[23] The list of names was collected from Wikipedia pages on the corresponding elections, which contained the names of candidates as used during the campaign cycle, as opposed to their birth name.[24]

[22] Available: `https://cloud.google.com/video-intelligence/` (Accessed March 24, 2019)

[23] We provided names for Republican or Democratic candidates only, except for the senatorial elections in Maine and Vermont, which featured popular independent candidates.

[24] For example, Bernie Sanders versus Bernard Sanders.

Although the literature suggests that these ASR systems can achieve very low error rates (Prabhavalkar *et al.*, 2017), the benchmark datasets used in those studies may not be comparable to our campaign ad dataset. In attempt to resolve this gap, Proksch *et al.* (2019) analyze political speech in EU State of the Union debates and show that using the GCP's auto-generated transcripts for bag-of-words text models is comparable to using the human-annotation. We also find GCP's algorithm to be accurate and suitable for our task. The most common error is mixing up similar-sounding words, but these errors do not diminish the substantive meaning of the text. Figure 7 shows an example of a transcript obtained using the video intelligence API and its ground truth, illustrating the accuracy of the GCP's transcription.

3.2.2 Text features

We use the transcripts generated from Google's ASR algorithm to perform several video classification tasks, i.e., issue mentions (Section 4.1), opponent mentions (Section 4.2), and sentiment analysis (Section 4.5). For the first two tasks, we employ keyword-based methods using only the raw transcripts, but for sentiment analysis, we construct text features and train a classifier. In this section, we describe the procedure for computing these text features used for sentiment analysis.

We use a bag-of-words for our sentiment analysis. We chose this feature over more advanced alternatives, such as sentence embeddings, because the computer-generated contained no punctuation, which would have severely degraded the performance of any syntax-based method. Instead, we focus on using natural language processing (NLP) techniques to preprocess and filter the text, producing a concise vocabulary for computing a vector of word counts. We achieve this using the Python package `spacy`.[25]

Figure 8 illustrates this pre-processing procedure. We begin by annotating the raw text with part-of-speech (POS) tagging and named entity recognition (NER), separating the named entities from the rest of the text. This step assigns part-of-speech tags (e.g. adjective, noun, adverb, etc.) to each word in the transcript and detects and labels n-grams corresponding to named entities

[25]Available: `https://spacy.io/`. (Accessed March 26, 2019)

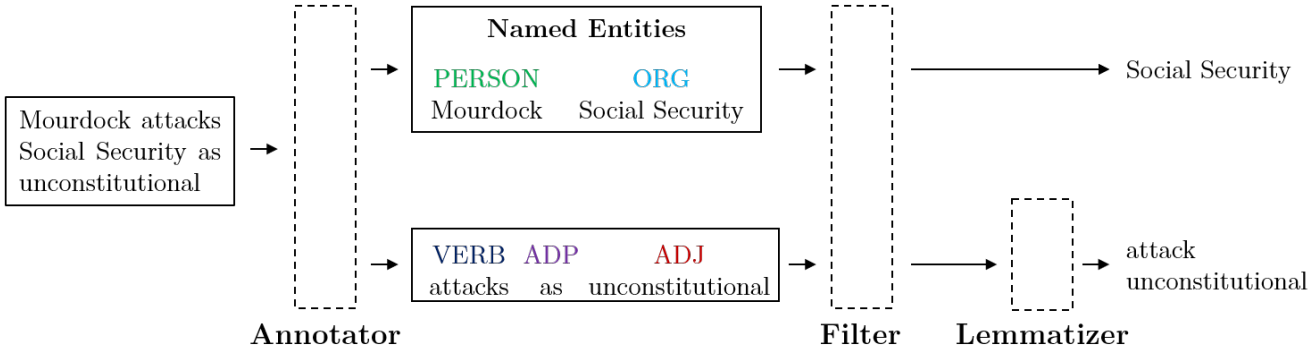


Figure 8: Procedure for Tokenizing the Auto-generated Transcript to Extract the Text Features. The input text is an excerpt from a transcript for a campaign ad for Democratic candidate Joe Donnelly in the 2012 Senatorial election in Indiana. First, an annotator labels the text with part-of-speech (POS) and gathers the named entities. The filter removes named entities labeled as people and some uninformative stop words from the non-entity group. The non-entities are then passed through a lemmatizer to recover the root form of each word. The resulting tokens are then gathered together to be used as input for the bag-of-words feature constructor.

(e.g. person, organization, location) using LSTM neural networks (Dozat and Manning, 2016; Andor *et al.*, 2016). We are particularly interested in recognizing named entities because it allows us to tokenize the sentence without splitting up n-grams that should be kept together, such as Social Security or Affordable Care Act.

After annotating the text, we filter out tokens that carry no meaning in regards to the sentiment of the ad. In particular, we remove named entities corresponding to people, with the exception of Barack Obama, Ronald Reagan, and Nancy Pelosi, because these names were often used to emphasize a candidate's or opponent's ideology. This step serves primarily to remove the multiple references to the candidate and the opponent, which do not carry information about the sentiment of ads. For the non-entities, we use conservative stop-word removal.[26] The reason for minimal removal is that some prepositions (e.g over, under), modifiers (e.g. very, few, most), and verbs (e.g. can, cannot, do) are informative of the connotation of an ad.

Lastly, the filtered tokens not corresponding to entities are passed through a lemmatizer to recover the dictionary form (i.e., the basic form found in the dictionary) in order to condense similar words into a single token. Lemmatization is an alternative to stemming that uses POS

[26]The stop-word list is a modified version of `https://www.ranks.nl/stopwords` (Accessed March 26, 2019)

tags as well as the context of a word within a sentence to determine its dictionary form. This contrasts with heuristic truncation, which can transform or erase the original meaning of the word. For example, the word "caring" is truncated to "car" under conventional stemmers, but a lemmatizer would correctly reduce this to "care". Lemmatizers can also handle words with unusual plural forms, such as "mouse" and "mice." Stemming would treat these words separately, while a lemmatizer would group them both under "mouse." After lemmatization, we collect the resulting tokens and compute the counts to produce the bag-of-words vector, using a dictionary constructed from the entire corpus of tokenized video transcripts.

3.2.3 Music features

In addition to text, the type of the music used in the video also provides useful information about the tone or purpose of the given ad. Hence we apply the algorithm of Ren *et al.* (2015) to compute features that are useful for classifying the types of the music (see references therein for other algorithms).[27]

Like the spectral fingerprinting method described in Section 2.2, the music features are based on the spectrogram of the audio signal. We first obtain the audio signals from the video.[28] Before the FFT step in computing the spectrogram, each frame is passed through a pre-emphasis filter, which boosts the high-frequency components.[29] This operation helps to distinguish high-frequency components of a signal from noise in the spectrum that tend to have weaker magnitude than the low-frequency components. After pre-emphasis, we follow the same procedure as the one used in Section 2.2; the frames are weighted by a Hamming window, and the 1024-point FFT, followed by the computation of an absolute value producing the spectrogram.[30]

[27]We choose this algorithm based on its performance in the Music Information Retrieval Evaluation eXchange (MIREX). The MIREX is a community-driven framework for the scientific evaluation of systems and algorithms for various problems in the domain of machine learning in audio and music information retrieval Downie (2008). MIREX runs an annual contest over multiple music classification tasks to evaluate their performances.

[28]Following Ren *et al.* (2015), we sample at the rate of $F_s = 22,050$kHz and compute the one-sided spectrogram using $W = 1,024$ (or 46 ms)-length frames with an overlap factor of $1/2$.

[29]For a given framed segment of W samples $x[w]$, where w denotes the wth sample, the pre-emphasized segment is computed as $y[w] = x[w] - 0.95x[w-1]$, defining $x[w] = 0$ for $w < 0$.

[30]Hence, the number of frequency bins is fixed to $K = 513$, yielding the matrix of spectrogram $S \in \mathbb{R}^{N \times K}$, where

The music features consist of several characterizations of spectrogram with the goal of capturing the different perceptual qualities of the audio, such as pitch timbre, tempo, or rhythm. We can categorize these features as *short-term* or *long-term*. Here, we provide a high-level description of these features, leaving the mathematical description to Appendix A.3.

The short-term features are used to quantify the timbral[31] qualities of the audio signal and are characterized by the shape of the spectrum. The first short-term feature we use is a statistical spectrum descriptor that summarizes the shape of spectrum across all frames in the spectrogram. Specifically, we compute the spectral centroid, skewness, kurtosis, rolloff, flux, for each frame in the spectrogram. The flux measures how quickly the spectrum is changing in from frame-to-frame, while the other statistics describe the distribution of frequencies spectrum. Computing these statistics for all N frames gives an $N \times 5$ feature matrix.

The second short-term features calculated are the mel-frequency cepstral coefficients (MFCC), a popular feature commonly used in speech classification tasks that has recently seen success in music-related tasks. Like the statistical spectrum descriptor described above, the MFCC characterizes the shape of the magnitude spectrum. However, instead of using summary statistics, it computes a low-dimensional projection of the spectral energy. Following the suggestion of Ren *et al.* (2015), we compute the MFCC for each frame in the spectrogram and retain the first 20 coefficients. We also use the energy of each frame as an additional feature. This computation yields an $N \times 21$ feature matrix.

The third short-term feature we compute is called the octave-based spectral contrast, which characterizes variations between the peaks and valleys in 8 logarithmically-spaced subbands of the spectrum. In general, the peaks correspond to tonal-like signals while the valleys correspond to noise-like signals. The contrast, defined as the difference between the peaks and valleys, reflects the relative strength of tonal signals. We concatenate the valleys and the contrast together to

N is the number of frames in the spectrogram.

[31] Timbre is the perceived sound quality that distinguishes between different sources of sound production, such as a violin or a piano.

form an $N \times 16$ feature matrix.

With the same frequency subbands used in computing the octave-based spectral contrast, we also compute the spectral flatness measure (SFM) and spectral crest measure (SCM) for each frame and each band. The SFM is a measure of how dispersed the frequencies are in the the subband, while the SCM measures how extreme the extreme values in the subband are. Both of these measures quantify how noise-like or tone-like the signal is. Computing the SFM/SCM for all frames yields an $N \times 16$ feature matrix.

For each of these four short-term features, we summarize its temporal variation by a vector of mean and standard deviation across all frames. Putting all together yields a 116 dimensional vector as a summary of the distribution of the short-term features for the audio signal.

By computing the mean and standard deviation of the short-term features across all frames, we reduce dimensionality but lose temporal information, which is critical for characterizing perceptual sound qualities like melody, tempo, and rhythm. To address this issue, we compute the modulation feature spectrogram, which characterizes the frequency content of each of the feature dimensions as it varies throughout the signal. We then construct a septal-based spectral contrast feature to characterize the shape of the spectrum for each frame in the feature spectrogram. These features are computed for each of the four short-term features, yielding a 296 dimensional feature vector (see Appendix A.3.2 for details).

The final feature we use is called the joint-frequency feature, which is computed from the joint-frequency spectrogram of the audio signal. After computing the spectrogram, another Fourier transform is applied along each frequency bin to compute the joint-frequency spectrogram. Like the short-term feature, we characterize the joint-frequency spectrogram using OSC, SFM, and SCM, measuring the tonal- and noise-like qualities of the audio over different musical beat rates that are informative of the audio's tempo and rhythm. This step creates a 224 dimensional vector of features (see Appendix A.3.2 for details).

Combining the short-term and long-term features, we obtain a 636 dimensional feature vector

and use it for our music mood classification, described later in Section 4.4, and sentiment analysis, described in Section 4.5.

4 Validation Results

In this section, we present the validation results, evaluating the performance of automated coding against that of human coding.

4.1 Issue Mention

A key set of variables in the WAP/WMP data sets indicate whether or not an TV advertisement mentions or pictures certain political issues and actors of interest. These variables can be broadly classified into three categories. The first set of variables indicate whether each of ten prominent actors, including *Barack Obama*, *Nancy Pelosi*, *Mitch McConnell*, *Democrats*, and *Republicans*, are mentioned and/or pictured.[32] The second set of variables represent whether each of twelve politically-charged words or phrases, such as *Tea Party*, *Change*, *Conservative*, *Wall Street*, and *Big Government*, are mentioned.[33] The final set of variables indicate whether 61 issues, including *Tax*, *Jobs/Employment*, *Gun Control*, *Drugs*, and *Immigration*, are mentioned.[34]

We automate the coding of these variables by simply checking whether at least one keyword is mentioned in each video[35]. While the WMP codebook provides the issue labels (i.e., issues, names

[32]The actual question to human coders reads, "Are any of the following mentioned or pictured in the ad?" There are four possible responses; 0 = No, 1 = Yes (in a way to show approval or support), 2 = Yes (in a way to show disapproval or opposition), and 3 = Yes (unclear whether in support or opposition). For the purpose of our validation study, we focus on issue mentions by turning this and other similarly coded variables into binary variables.

[33]The actual question to human coders reads, "Are any of these words/phrases *specifically* mentioned in the ad?" (*italics* original) to which the possible response is either 0 = No or 1 = Yes. There are three extra words/phrases that only appear in the 2014 dataset: *Working Class*, *Middle Class*, and *Upper Class/Wealthy*. We exclude these three variables from our analysis for consistency.

[34]The actual question to human coders reads, "Are any of the following issues mentioned in this ad?" and requires either 0 = No or 1 = Yes as an answer. The WMP data have a total of 63 issues. We decided to merge "abortion" and "women's health" into one issue for the sake of consistency.

[35]The comparability between the original and automated codings, while not perfect, is further enhanced by the fact that the WMP also codes an issue as being mentioned even if it was not addressed in a policy context (e.g., a "self-made businessman" counts as a mention of *business*(Wesleyan Media Project, 2016a)), which creates a degree of similarity to our keyword-based algorithm.

		Automated coding			
		Audio Data Only		Audio and Visual Data	
		No	Yes	No	Yes
WMP coding	No	197,986 (95.72%)	1,501 (0.73%)	197,173 (95.33%)	2,314 (1.12%)
	Yes	1,776 (0.86%)	5,573 (2.69%)	1,488 (0.72%)	5,861 (2.83%)

Table 3: Comparison of the Issue Mention Variable between the WMP and Automated Codings. The value in each cell corresponds to the frequency of the four different combinations of results from the WMP and automated coding schemes. The first two columns show the results when we only use audio transcripts to automatically detect keywords, and the remaining two columns are for when both audio transcripts and image texts are used. There are a total of 206,836 video-issue pairs across 83 issues.

of political actors, phrases), they do not always constitute useful keywords. Thus, we choose a set of relevant words and phrases as keywords and in certain cases exclude the use of these keywords in irrelevant contexts.[36] It is important to note that we do not stem or lemmatize the transcripts and texts, as doing so would result in making subtle distinctions between uses of the same keywords and phrases.[37]

Among a total of 83 issues, we use the WMP issue names and the last names of political actors for 44 issues (e.g., "tax" for the *Tax* issue, and "Pelosi" as the *Pelosi* issue). For 16 issues, we also added synonyms and some words that share the same roots. For example, we add "Chinese" as another keyword for the *China* issue, and include the word root "agricult," in addition to "farm" as a keyword for the "farming" issue. Third, for 21 issues, we have added relevant common expressions. For example, we include "climate change" as an additional keyword for the *global warming* issue. Another example is the addition of "second amendment," "2nd amendment," "NRA," and "bear arms" for the *gun control* issue. Finally, for the *jobs/employment* and *abortion/women's health* issues, we use a more extensive list of relevant words and phrases. A complete list of keywords we use appear in Tables A.2 to A.4 of Appendix A.4.

Given the set of keywords, we then code an issue mention variable as 1 if at least one of the

[36] One example of such restriction is ignoring "Wall Street Journal" in looking for mentions of *Wall Street*.

[37] For example, stemming would make it impossible to distinguish "He is not doing a good job as our Governor" and "We must stop importing our jobs overseas" in their uses of the word "job."

keywords is either mentioned in the audio transcript or pictured in the image texts extracted from each video file (and 0 otherwise).[38] We first examine the overall level of agreement between the WMP issue mention variables and our automated coding of them across 206,836 ($= 2,492 \times 83$) video-issue pairs. Table 3 shows that they agree in more than 98% of all video-issue pairs considered. If we were to assume that the WMP coding is actually correct, then we would have a false positive rate (i.e. the proportion of Yes for the automated coding among the cases the WMP codes No) of 1.12% and false negative rate (i.e., the proportion of No by automated coding among the cases the WMP codes Yes) of 0.72%. The use of image texts to complement audio transcript results in 1,101 additional positive findings. The WMP codes 288 of them as Yes, and the remaining 813 as No. While this might appear to be decreasing the accuracy of the coding algorithm, it turns out that the WMP coding in these cases is often incorrect as shown in a verification study via the Amazon Mechanical Turk below.[39]

Although the overall agreement rate is high, this is in part due to the fact that many values of issue mention variables are zero: each video only mentions a small number of issues. In addition, it is possible that both humans and machines may incorrectly code some variables. Hence we use the Amazon Mechanical Turk to further validate both automated and manual codings. To do this, we randomly sample 700 video-issue pairs such that 200 pairs represent the cases of agreement whereas the remaining 500 pairs correspond to the cases of disagreement with 325 "false positives" (i.e., automated coding gives Yes while human coding gives No) and 125 "false negatives" (i.e., automated coding gives No while human coding gives Yes). We further stratify on the WMP issue categories so that the distribution of WMP issue category in the selected sample closely mirrors that of the entire set of pairs. Each issue-video pair is reviewed by five different individuals who

[38]There are two exceptions. First, for *Congress* and *Wall Street* issues, we exclusively use audio transcripts only because on-screen text frequently contained mentions of "[candidate name] for Congress" and citations from "Wall Street Journal" that increased false positive declarations. Second, for *Barack Obama*, we also incorporate facial recognition algorithm to complement the transcripts and text, thus coding the variable as 1 if Obama was either mentioned or pictured. This allows for a better approximation of the original WMP variable.

[39]For a sample of 39 videos for the *business* issue in the false positive condition, for example, MTurkers agree with the automated coding in 28 or about 72% of the cases. For a sample of 40 videos for the *Congress* issue in the false negative condition, MTurkers agree with the automated coding in 29 or about 73% of the cases.

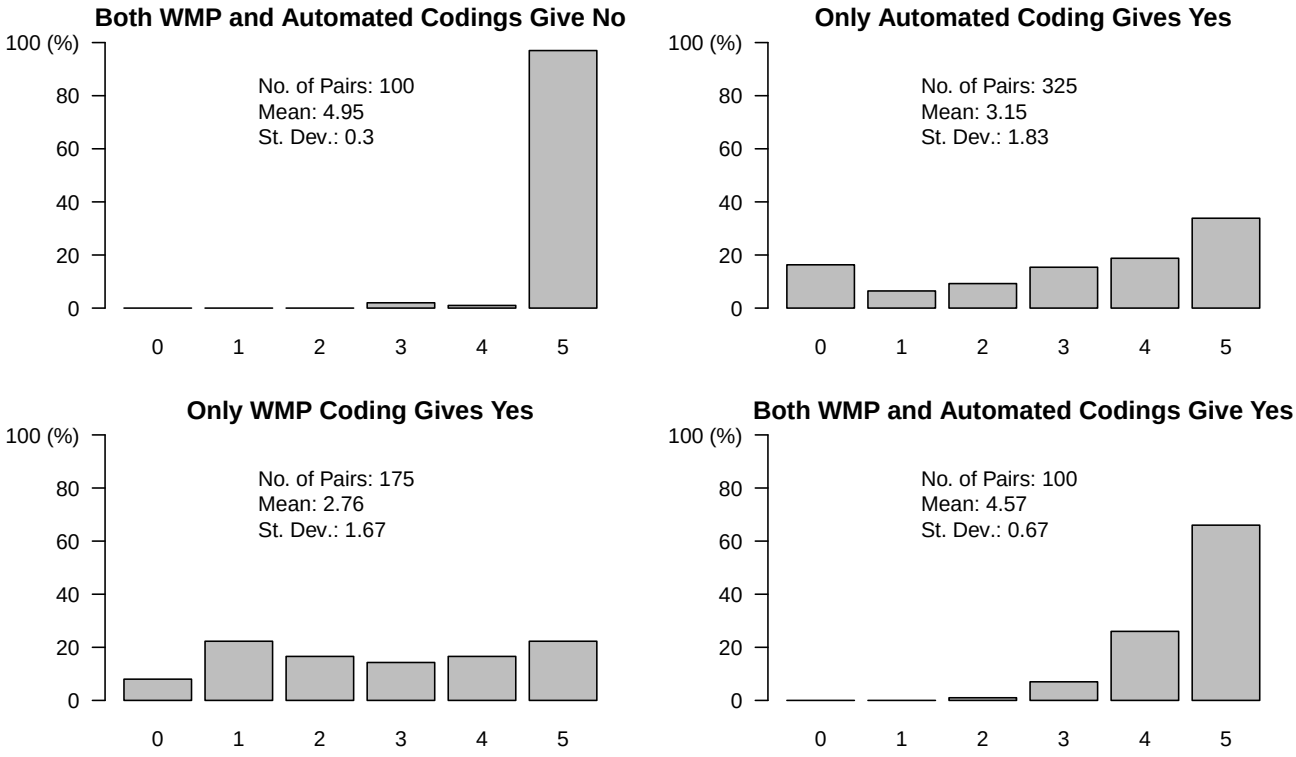


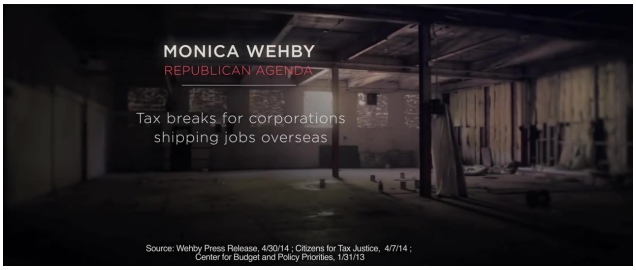
Figure 9: Number of MTurkers Who are in Agreement with Automated Coding For Issue Mentions. The four cases are based on the agreement and lack thereof between the WMP and automated codings. A total number of MTurkers for each task (hence the maximum possible number of MTurkers who agree with automated coding) is five. The texts within each plot show the number of issue-video pairs included in each sample as well as the mean and standard deviation of the number of MTurkers in agreement with the automated coding.

were granted the qualification of MTurk Masters. An example script including instructions as seen by MTurkers can be found in Figure A.5 of Appendix A.5.[40]

Figure 9 presents a bar plot of the number of MTurkers who agree with the automated coding in each of the four different conditions. When the WMP and automated codings agree with each other (the two diagonal plots), many MTurkers reach the same judgment, as can be seen by the numbers concentrated tightly around 4 or 5. When the WMP and automated codings disagree with each other (off-diagonal plots), we find that the MTurkers often disagree with one another and are slightly more likely to agree with the automated coding. The results imply that the automated

[40]To be fair to MTurkers, we set the average monetary rewards to be 9 dollars per hour, assuming that each MTurker viewed a single issue-video pair twice and used an additional 30 seconds to record their responses. This led to the payment of \$0.15, \$0.23, and \$0.38 for a single 15-, 30-, and 60-second-long video, respectively. A single MTurker could complete up to 50 issue detection tasks across any video lengths.

(a) Ad by Senator Mark Warner (Democrat, Virginia). The automated coding algorithm detected an entire excerpt from the newspaper in the background, incorrectly choosing the *tax* issue.



(b) Ad by Senator Jeff Merkley (Democrat, Oregon). The automated coding algorithm detected the word "Budget" from an organization quoted by name on the screen, thus incorrectly choosing the *budget* issue.

Figure 10: Examples Illustrating the Mistake of the Automated Coding for the Issue Mention Variable.

coding is at least as accurate as the WMP coding even in cases of disagreement between the two coding schemes.

Finally, we examine the conditions under which the automated procedure makes mistakes in coding the issue mention variable. To do this, we select a random subset of 200 issue-video pairs from the 500 disagreement cases used for the MTurk study (130 "false positives" and 70 "false negatives"). We then carefully watch these videos and establish the ground-truths for these pairs. We find that the automated coding made 40 mistakes out of 130 "false positive" evaluations, implying that the automated coding outperformed the WMP coding in these cases. The most frequent mistake by the automated coding (17 cases) is to pick up extra visual features. Figure 10a shows such an example, in which the automated coding algorithm detected the entire excerpt from a newspaper article pictured in the ads, thus contributing to a false positive declaration.[41]

The second most frequent mistake of our automated coding procedure is due to the fact that it ignores the context (15 cases). For example, the use of the keyword "energy" that refers to the personal quality of being energetic was considered as mentioning *energy*, whose definition in this case is geared towards physical resources. Another example is the incorrect classification of the keyword "change" from the phrase "life-altering change for children with autism," as mentioning

[41] In some cases, articles are sufficiently short and clear that this is not seen as a coding mistake.

the issue of *change*, whose original intention is to refer to a political reform. The automated coding also incorrectly treated the names of institutions as mentioning certain issues. One such example is given in Figure 10b, where the coding algorithm detected the keyword "budget" in the name of an organization, "Center for Budget and Policy Priorities," to whom a quote is attributed. These represent difficult cases for automated coding algorithms, which careful human coders will have a better chance of correctly classifying.

We have also examined 70 "false negative" cases and found that the automated coding resulted in 33 mistakes, suggesting that the WMP and automated codings had a similar performance in these cases. The most frequent reason for mistakes (21 cases) is missing keywords, which is relatively easy to correct by simply including an additional set of keywords. Examples include "college debt" and "teachers" for the *education* issue and "regulatory burden" and "red tape" for the *government regulation* issue. The second most frequent reason is a subtle mentioning of issues. For example, an ad included the phrase "cracking down on China cheating", as an indirect reference to the *trade* issue; another ad used the phrase "borrowing from tomorrow and wasting it today," in the context of the *government spending* issue. These cases represent a challenge to keyword-based approaches such as ours, since accounting for all such variations in indirect references is impossible for most practical purposes.

4.2 Opponent Mention

The WMP data sets also include a separate variable that determines whether the ad mentions the opponent in the main part of an ad excluding the section corresponding to the oral approval.[42] Our automated coding algorithm is exactly the same as before, except that the keywords used are three different forms of the opponent's last name: last name itself, possessive, and possessive without an apostrophe. For example, if the name of the opponent is Jane Roe, then we use "Roe," "Roe's," and "Roes." We include the possessive form without an apostrophe in order to account

[42]The question to the human coders reads "Excluding the *oral approval*, is the opposing candidate mentioned by name in the ad?" (emphasis original) to which coders answer either 0 for No or 1 for Yes.

		Automated coding			
		Audio Data Only		Audio and Visual Data	
		No	Yes	No	Yes
WMP coding	No	1,273 (51.43%)	64 (2.59%)	1,260 (50.91%)	77 (3.11%)
	Yes	127 (5.13%)	1,011 (40.85%)	28 (1.13%)	1,110 (44.85%)

Table 4: Comparison of the Opponent Mention Variable between the WMP and Automated Codings. The value in each cell corresponds to the frequency of the four different combinations of results from the two coding schemes. The first two columns show the results when we only use audio transcripts to detect keywords, and the remaining two columns are for when both audio transcripts and on-screen text are used. A total of 2,475 videos were considered, after discarding 17 videos for which the WMP variable was missing.

for the instances where audio transcription or image text detection algorithm misinterprets the possessive as a plural and thus incorrectly suppresses the apostrophe.

Table 4 shows that the automated coding agrees with the original WMP coding in over 95% of the cases when both audio transcripts and on-screen text detection are combined. When compared to the case of the issue mention variable, we find that there is a better balance between positive and negative cases. A total of 105 disagreements are recorded, and about three quarters of them are found in the condition where the WMP coding returns No and the automated coding returns Yes. Using the image text detection to complement audio transcript boosted the performance of the coding algorithm, leading to an increase of 99 in the "true positive" condition and and increase of 13 in the "false positive" condition.

As before, we carefully watch all 105 videos in the disagreement conditions. Out of 77 "false positives" (i.e., the automated coding gives Yes while the WMP gives No), the automated coding makes three mistakes (3.9%), all of which are due to the algorithm picking up the opponent's last name in the background objects such as newspaper articles and posters (see one example shown in Figure A.7a of Appendix A.6). For the 28 videos in the "false negative" condition (i.e., the automated coding gives No while the WMP gives Yes), the automated coding makes 18 mistakes (64%). Based on the manually-assigned true labels for the 105 disagreement cases, the precision,

Figure 11: Montage of Collected Images of 75 Senate Candidates from the 2012 and 2014 Election Cycles. The candidates are arranged in the alphabetical order, from Todd Akin in the top left to Mark Zaccaria in the bottom right.

recall, and accuracy for the automated coding are 0.96, 0.80, and 0.79 respectively, while the same statistics are much lower for the WMP (0.68, 0.20, 0.21).[43] Thus, the performance of the automated coding exceeds that of the WMP human coding.

By far the most frequent reason (15 out of 18) why the automated coding fails to detect the mention of opponents is the mis-transcription of the last names of the opposing candidates, especially when they are relatively uncommon. Examples include mishearing "Tisei" as "to say," "Zeldin" as "Sheldon," "Lankford" as "Langford," and "Critz" to "Crits." Among the three remaining errors, one is because the opponent was mentioned by first name only, another because Romney referred to Obama as "a (potential) president," and the last one due to the last name only appearing as an on-screen text using handwritten fonts (see Figure A.7b of Appendix A.6).

4.3 Face Recognition

Another set of variables that are included in the WAP/WMP data set indicate whether the favored candidate or the opponent was pictured in the main part of an ad excluding the section

[43] The precision, recall, accuracy are defined as $\frac{\texttt{true positives}}{\texttt{true and false positives}}$, $\frac{\texttt{true positives}}{\texttt{true positives and false negatives}}$, $\frac{\texttt{true positives and negatives}}{\texttt{total}}$, respectively.

corresponding to the oral approval.[44] We combine this variable with another indicator that codes whether the favored candidate is seen *directly* speaking to the audience during the oral approval section.[45] Replication of this combined variable is not straightforward, however, since it is coded as No even when the ad contains images of the candidates if they do not directly address the audience.[46] To more closely replicate this variable, we must identify the oral approval segment. For this validation, however, we evaluate the ability of the face recognition algorithm to correctly detect candidate faces regardless of which segments picture the candidates.

For illustration, we automatically code these variables for a total of 75 Senate Candidates in the 2012 and 2014 election cycles for whom we have at least one YouTube video matched to a WMP video file. We first collected the images of these candidates by automatically scraping the Wikipedia pages dedicated to each. In cases where the pictures were either missing, obscured, or severely outdated, these were replaced by manual searches on the Internet. Figure 11 presents a montage of all images for these candidates. Finally, we apply the face recognition algorithm described in Section 3.1.3, declaring a match when the Euclidean distance between a detected and opponent's reference image is below 0.9.[47]

Table 5 shows the patterns of agreements and disagreements between the WMP and automated codings of the candidate appearance variables. For favored candidates, the two modes of coding agree in about 78% of the cases. Among the remaining cases of disagreement, the automated face recognition returns Yes in about two-thirds of them. The overall agreement rate is higher for the opposing candidates, with the two agreeing on roughly 90% of the cases. Most of the disagreements correspond to the cases in which the face recognition algorithm returns No.

[44]The question posed to the coders reads, "excluding the *oral approval*, is the favored candidate / opposing candidate pictured in the ad?" to which possible responses are 0 for No and 1 for Yes.

[45]The question reads "Does the candidate physically appear on screen and speak to the audience during oral approval?" to which the possible answers are 0 for No and 1 for Yes.

[46]The definition imposed by the WMP requires the candidates themselves to be seen uttering the line "I'm [candidate name] and I approve this message." This does not include the cases where the picture of the candidate appears and the oral approval is given in voiceovers.

[47]This choice of threshold reflects the suggestion of Schroff *et al.* (2015) for a value around 1 and our verification procedure with the images of Ronald Reagan illustrated in Section 3.1.3, in which 0.9 kept both the false positive and false negative case to just one each.

		Automated coding			
		Favored candidate		Opposing candidate	
		No	Yes	No	Yes
WMP coding	No	58 (7.56%)	109 (14.21%)	490 (63.89%)	12 (1.56%)
	Yes	57 (7.43%)	543 (70.80%)	65 (8.47%)	200 (26.08%)

Table 5: Comparison of Candidate Appearance Variables between the WMP and Automated Face Recognition. The value in each cell corresponds to the frequency of the four different combinations of results from the WMP and automated coding schemes. The first two columns are for the appearance of the favored candidates, and the remaining two columns are for the appearance of opposing candidates. Total sample size is 767.

We watch all 243 videos of disagreement cases (166 for the favored candidates and 77 for the opponents) to identify the likely reasons why the WMP coders and the face recognition algorithm reach different conclusions. For the cases of favored candidates, the primary reason for disagreement (94 cases or 57%) is that the algorithm has detected the images of favored candidates within the oral approval segment.[48] This is expected as we did not add any filter regarding the restriction made by the WMP. The second most frequent reason (48 cases or 29%) is that the algorithm had a difficult time dealing with angled, occluded, and dimly-lit images. The remaining 24 disagreements (14%) were due to the mislabeling by the WMP coders. For the opposing candidates, 51 disagreements (66%) are due to the angle, lighting, and quality of the images. The remaining 26 cases (34%) were mislabeling on the part of the WMP. Two illustrative examples (angled and processed images) of inaccurate automated coding appear in Figure A.8 of Appendix A.6.

Finally, we evaluate the performance of the face recognition algorithm alone by removing the restriction that the favored candidates appear in the main segment of an ad alone. To do this, we manually recode the WMP variables for the disagreement cases so that the variable represents whether the favored candidate appears in any segment. We also correct the labeling mistakes made by the WMP coders. Under the assumption that the agreements between the manual and automated codings indicate the accurate classification, we compute the precision, recall, and

[48]This restriction appears to be confusing for the WMP coders who also made the same mistake in some cases.

accuracy of the face recognition algorithm. We find that they are 1.00, 0.91, and 0.92 for the favored candidates and 1.00, 0.81, and 0.93 for the opposing candidates, respectively. These numbers represent an impressive performance of the face recognition algorithm.

4.4 Music Mood Classification

Many political campaign ads have background music. The WAP/WMP data set contains three dichotomous variables that describe its mood as "ominous/tense," "uplifting," and "sad/sorrowful."[49] The WMP coding of these three categories of music are not mutually exclusive: among 2,276 videos for which at least one type of music is indicated, 353 or about 15% of them have more than one category selected by the coders. The three classes are relatively imbalanced as well: "uplifting" music was most frequently coded with 1,586 videos (70%), followed by "ominous/tense" music with 725 videos (32%), and "sad/sorrowful" music with 320 videos (14%).

For this music mood classification task, we take a supervised learning approach. Specifically, we train an SVM classifier with a radial basis function and hyperparameters tuned via 5-fold cross-validated grid search with the loss function optimized for F_1-score. The music features used for classification are described in Section 3.2.3, and we randomly select 440 videos (about 15% of the data set) to which the WMP has given at least one of the three labels as the test set.[50] The music encoding is done independently for each category. While this allows multiple categories of music to be selected, it should be noted that it does not take into account the negative correlation between the WMP encodings of these variables.

Table 6 presents the frequency of agreements and disagreements between the original WMP and automated codings. Overall, the average rate of agreement is similar across the three types of music: 78% for "ominous/tense" and "uplifting" music, and 83% for "sad/sorrowful" music.

[49]The original question to the coders is "if music is played during the ad, how would it best be described?" to which the possible answers were 0 for No and 1 for Yes to each of "ominous/tense music," "uplifting music," "sad/sorrowful music," and "another type of music." If the coders chose the last, they were also asked to describe in words what specific type of music they referred to. We ignore the last category for our analysis.

[50]We found several cases where two or more WMP coders disagreed on the coding of the variables. We remove such cases from the data since the training of the SVM classifier requires the labels in the training set to be fixed to single values.

		Automated coding					
		Ominous/Tense		Uplifting		Sad/Sorrowful	
		No	Yes	No	Yes	No	Yes
WMP coding	No	237 (53.86%)	64 (14.55%)	66 (15.00%)	65 (14.77%)	334 (75.91%)	45 (10.23%)
	Yes	35 (7.95%)	104 (23.64%)	31 (7.05%)	278 (63.18%)	31 (7.05%)	30 (6.82%)

Table 6: Comparison of the Music Mood Variables between the WMP and Automated Codings. The value in each cell corresponds to the frequency of the four different combinations of results from the WMP and automated coding schemes. The three two-by-two matrices correspond to the three different moods of music used in the WMP data. The results shown here are from the test data set of size 440.

As alluded to above, the frequency of each type of musical moods is different, with the uplifting music being detected by both coding schemes in the majority of the sampled videos while the corresponding numbers are lower for "ominous/tense" and "sad/sorrowful" music. Among the disagreement cases, the classifier is more likely to return Yes compared to the WMP coding.[51]

Compared to the other classification tasks, the rate of agreement with the original WMP coding is lower for this musical mood classification task even though we use a supervised learning approach here. This is because classifying music into a specific type is known to be a difficult task, with the current state-of-the-art machine-learning approaches achieving about 70% classification accuracy for the benchmark MIREX dataset (Ren *et al.*, 2015). However, even human coders who generally achieve better classification accuracy do not come to complete agreement as to the type of music a given ad contains. The WMP reports the intercoder agreement rate of 83.9, 88.8, and 89.5% for the "ominous/tense," "uplifting," and "sad/sorrowful" categories, respectively, based on the 903 double-coded samples from the 2012 election cycle.[52]

Given the inherent difficulty of establishing the ground-truth for this variable, we use the Amazon Mechanical Turk as another source of coding. We assign five different MTurk coders with Masters qualification to each of the 440 videos included in the validation data set and ask them

[51] We obtain a higher rate of agreement when we combine the ominous/tense and sad/sorrowful types of music together and perform binary classification against uplifting music, although the difference in performance is modest.

[52] The corresponding numbers for the samples of the 2014 election cycle are 89.0, 89.8, and 91.5% among 1,939 videos.

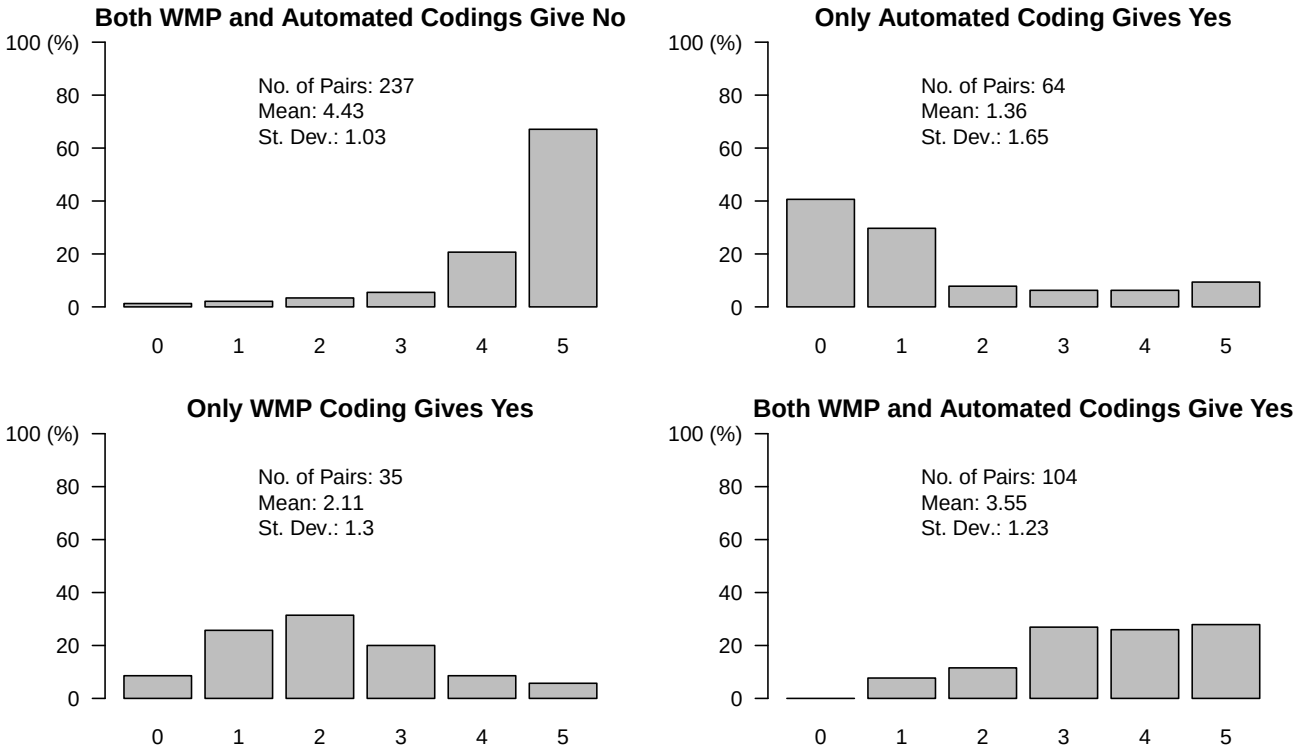


Figure 12: Number of MTurkers Who are in Agreement with Automated Coding for Ominous/Tense Music. Four cases are based on the agreement and lack thereof between the WMP and automated codings. A total number of MTurkers for each task (hence the maximum possible number of MTurkers who agree with automated coding) is five. The texts within each plot show the number of issue-video pairs included in each sample as well as the mean and standard deviation of the number of MTurkers in agreement with the automated coding.

to code whether the ad contains any of the three types of music. An example script along with instructions as seen by MTurkers can be found in Figure A.6 of Appendix A.5.[53] Figure 12 shows the distribution of the number of MTurkers in agreement with the automated coding under the four patterns of agreement for the "ominous/tense" category (corresponding plots for the other two categories can be found in Figures A.10 and A.11 in Appendix A.7). The figure suggests that the MTurkers are likely to agree when both the WMP and automated codings return the same result, although the responses are much more evenly distributed for the cases where both codings give "Yes." In the cases where the two codings disagree with each other, the MTurkers are more inclined to agree with the WMP coding, especially when only the automated coding returns "Yes."

[53]The monetary incentives are identical to the other MTurk classification task described in Section 4.1.

This suggests that the performance of the automated mood classifier may be falling short of the human standards for music mood classification.

The under-performance of automated coding is expected given the difficulty of this task. When we aggregate the MTurk coder responses into a binary variable based on the majority opinion in each case, the rates of agreement with the WMP coding are about 85% for all three categories (see Table A.5 in Appendix A.7), which are similar to the intercoder agreement rate reported by the WMP. This suggests that even human coders have a difficulty in some cases.[54]

4.5 Negative Advertisement

As a final validation exercise, we automatically detect the negativity of campaign ads. The WAP/WMP data contain two variables that are related to negativity. One classifies the tone of the advertisement as "positive," "negative," or "contrast." This variable is provided by the CMAG and thus is not coded directly by the WMP. The other variable, which is coded by the WMP research team, classifies the purpose of the advertisement as "contrast," "promote," or "attack."[55] In this paper, we automate the former variable, which better conforms to a classical definition of sentiment analysis.[56] We removed all ads labeled as "contrast" for simplicity, as they will have a mixture of both positive and negative contents. The remaining videos were assigned labels of 1 if positive and -1 if negative.

We examine three different models: Naive Bayes classifier, linear SVM, and non-linear SVM with radial basis function. For the inputs to these classifiers, we use three sets of features: text counts as described in Section 3.2.2, music feature vector as described in Section 3.2.3, and a combined feature formed by concatenating the text and music features. In constructing the text

[54]Figure A.9 of Appendix A.7 illustrates that for 31 videos in which the WMP coders themselves were split in assigning labels, MTurk coders could not reach consensus either, further demonstrating the difficulty of the task.

[55]The original question to the coders reads, "in your judgment, is the primary purpose of the ad to promote a specific candidate, attack a candidate, or contrast the candidates?" to which possible responses are "contrast," "promote," "attack."

[56]Although these two variables appear to be quite similar, the correlation between the two are not perfect. For example, the correlation between "positive" and "promote" is 0.92 while the same statistic between "negative" and "attack" is 0.63.

		Automated coding					
		Text Only		Music Only		Text and Music	
		Negative	Positive	Negative	Positive	Negative	Positive
WMP coding	Negative	291 (56.18%)	34 (6.56%)	255 (49.23%)	70 (13.51%)	290 (55.98%)	35 (6.76%)
	Positive	43 (8.30%)	150 (28.96%)	63 (12.16%)	130 (25.10%)	39 (7.53%)	154 (29.73%)

Table 7: Comparison of the Ad Negativity Variable between the WMP and Automated Codings Using the Linear SVM. The value in each cell corresponds to the frequency of the four different combinations of results from the WMP and automated coding schemes. The three two-by-two matrices correspond to the three types of input data used to train the models. The size of this test set is 518, or roughly 20% of the full data.

counts, we use the Naive Bayes classifier on the validation set to optimize the following parameters: (1) the minimum term frequency to be included in the input feature, (2) whether or not the named entities are removed, and (3) the list of stopwords to be removed.[57] The text count vectorizer optimized by this procedure removes words occurring in less than 2% of the documents as well as named entities and uses our conservative stopword list described in Section 3.2.2. We then trained the three models on a random sample consisting of 80% of the WMP data entries and left the remaining 20% as the test set. The tuning parameters for each of the classifiers are optimized through three-fold cross-validation by maximizing F1 score over a dense grid of values. For classifiers which rely on distance measures between samples for classification (SVM), we standardize the features prior to the training procedure.

Table 7 shows how many times the original WMP codings and our automated codings agreed and disagreed when we use linear SVM which had the highest overall agreement among the three models used. Similar results for non-linear SVM and Naive Bayes classifier are found in Table A.6 of Appendix A.7. The results suggest that text features are effective for classifying ads into positive and negative ones, achieving the agreement rate of 85%. There is a reasonable balance between "false positives" and "false negatives." As expected, music features are less effective, yielding only the agreement rate of 74%. Combining text features with music features does not significantly

[57]The three candidates were our conservative stopword list, the default list of stopwords used by `sklearn`, and no stopword removal.

improve the performance of text features alone. We speculate that this is due to the fact that the music features we use are not designed specifically for detecting dark music often used for negative ads. A future research may consider new features that are targeted at this task. Nevertheless, the relatively high rates of agreement between the automated and human codings suggest that machine coding can be used to effectively classify positive and negative ads.

5 Concluding Remarks

We have shown that many of the variables from the Wisconsin Advertisement Project (WAP) and the Wesleyan Media Project (WMP) can be automatically coded by state-of-the-art machine learning methods without sacrificing accuracy. We believe that the use of automated coding will significantly improve the efficiency and scale of research on political advertisements. In particular, we hope that the methods used in this paper can shorten the time gap between an election cycle and the availability of high-quality data amenable to various content analysis. A similar machine learning approach can also be used in other areas of social science research as more video data become publicly available at YouTube and elsewhere on the Internet.

Over the last two decades, the WAP and WMP provided valuable resources for scholars who study political campaigns. As shown in this paper, these data sets also serve as ideal training and test data sets for researchers who wish to investigate the accuracy of cutting-edge machine learning methods. The performance of the automated coding algorithms in replicating the human-coded variables from the WAP and WMP varies from one task to another, and researchers can uniquely adapt the algorithms to maximize the quality of automated coding depending on their specific research agenda. For some of the tasks such as music mood classification where machines typically struggle, it might be necessary for the researchers to obtain a larger size of training data than is considered here or supplement the automation with the use of inexpensive human coding such as the one available via Amazon Mechanical Turk.

References

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally normalized transition-based neural networks. Tech. rep., arXiv:1603.06042.

Ashworth, S. and Clinton, J. D. (2006). Does advertising exposure affect turnout? *Quarterly Journal of Political Science* **2**, 27–41.

Banda, K. K. (2015). Competition and the dynamics of issue convergence. *American Politics Research* **43**, 821–845.

Baskurt, K. B. and Samet, R. (2019). Video synopsis: A survey. *Computer Vision and Image Understanding* **181**, 26–38.

Borrell Associates (2017). The final analysis: What happened to political advertising in 2016 (and forever).

Brader, T. (2005). Striking a responsive chord: How political ads motivate and persuade voters by appealing to emotions. *American Journal of Political Science* **49**, 388—-405.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. (2018). Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, 67–74. IEEE.

Chakraborty, S., Tickoo, O., and Iyer, R. (2015). Adaptive keyframe selection for video summarization. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, 702–709. IEEE.

Clinton, J. D. and Lapinski, J. S. (2004). "targeted" advertising and voter turnout: An experimental study of the 2000 presidential election. *Journal of Politics* **66**, 69–96.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, vol. 1, 886–893. IEEE Computer Society.

Dietrich, B. J. (2018). Using motion detection to measure social polarization in the u.s. house of representatives. Tech. rep., Department of Political Science, University of Iowa.

Dietrich, B. J., Enos, R. D., and Sen, M. (2018). Emotional arousal predicts voting on the u.s. supreme court. *Political Analysis* **27**, 2, 237–243.

Downie, J. S. (2008). The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology* **29**, 4, 247–255.

Dozat, T. and Manning, C. D. (2016). Deep biaffine attention for neural dependency parsing. Tech. rep., arXiv:1611.01734.

Fridkin, K. L. and Kenney, P. (2011). Variability in citizens' reactions to different types of negative campaigns. *American Journal of Political Science* **55**, 307–325.

Gerber, A. S., Gimpel, J. G., Green, D. P., and Shaw, D. R. (2011). How large and long-lasting are the persuasive effects of televised campaign ads? results from a randomized field experiment. *American Political Science Review* **105**, 135–150.

Goldstein, K. and Freedman, P. (2002). Campaign advertising and voter turnout: New evidence for a stimulation effect. *Journal of Politics* **64**, 721–740.

Haitsma, J. and Kalker, T. (2002). A highly robust audio fingerprinting system. In *Ismir*, vol. 2002, 107–115.

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE* **66**, 1, 51–83.

Huber, G. A. and Arceneaux, K. (2007). Identifying the persuasive effects of presidential advertising. *American Journal of Political Science* **51**, 961–981.

Jin, X. and Tan, X. (2017). Face alignment in-the-wild: A survey. *Computer Vision and Image Understanding* **162**, 1–22.

Kang, T., Fowler, E. F., Franz, M. M., and Ridout, T. N. (2017). Issue consistency? comparing television advertising, tweets, and e-mail in the 2014 senate campaigns. *Political Communication* **35**, 32–49.

Kaplan, N., Park, D. K., and Ridout, T. N. (2006). Dialogue in american political campaigns? an examination of issue convergence in candidate television advertising. *American Journal of Political Science* **50**, 724–736.

Knox, D. and Lucas, C. (2018). A dynamic model of speech for the social sciences. Tech. rep., Department of Politics, Princeton University.

Krupnikov, Y. (2011). When does negativity demobilize? tracing the conditional effect of negative campaigning on voter turnout. *American Journal of Political Science* **55**, 797–813.

Lee, C.-H., Shih, J.-L., Yu, K.-M., and Lin, H.-S. (2009). Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features. *IEEE Transactions on Multimedia* **11**, 4, 670–682.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, 3730–3738.

Meirick, P. C., Nisbett, G. S., Harvell-Bowman, L. A., Harrison, K. J., Jefferson, M. D., Kim, T.-S., and Pfau, M. W. (2018). To tell the truth: Ad watch coverage, ad tone, and the accuracy of political advertising. *Political Communication* **35**, 450–469.

Oppenheim, A. V., Willsky, A. S., and Nawab, S. (1996). *Signals and Systems (Prentice-Hall signal processing series)*. Prentice Hall Englewood Cliffs, New Jersey.

Prabhavalkar, R., Rao, K., Sainath, T. N., Li, B., Johnson, L., and Jaitly, N. (2017). A comparison of sequence-to-sequence models for speech recognition. In *Interspeech*, 939–943.

Proksch, S.-O., Wratil, C., and Wäckerle, J. (2019). Testing the validity of automatic speech recognition for political text analysis. *Political Analysis* 1–21.

Ren, J.-M., Wu, M.-J., and Jang, J.-S. R. (2015). Automatic music mood classification based on timbre and modulation features. *IEEE Transactions on Affective Computing* **6**, 3, 236–246.

Schaffner, B. F. (2005). Priming gender: Campaigning on women's issuesin u.s. senate elections. *American Journal of Political Science* **49**, 803–817.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 815–823.

Sides, J. and Karch, A. (2008). Messages that mobilize? issue publics and the content of campaign advertising. *Journal of Politics* **70**, 466–476.

Soltau, H., Liao, H., and Sak, H. (2016). Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. Tech. rep., arXiv:1610.09975.

Spiliotes, C. J. and Vavreck, L. (2002). Campaign advertising: Partisan convergence or divergence? *Journal of Politics* **64**, 1, 249–261.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Torres, M. (2018). Give me the full picture: Using computer vision to understand visual frames and political communication. Tech. rep., Department of Political Science, Washington University in St. Louis.

Wang, A. *et al.* (2003). An industrial strength audio search algorithm. In *Ismir*, vol. 2003, 7–13. Washington, DC.

Wang, M. and Deng, W. (2018). Deep face recognition: a survey. Tech. rep., arXiv:1804.06655.

Wesleyan Media Project (2016a). 2016 political ad coding supplemental guide. Tech. rep.

Wesleyan Media Project (2016b). Political advertising in 2012 codebook, ver 1.1. Tech. rep.

Wesleyan Media Project (2017). Political advertising in 2014 codebook, ver 1.0. Tech. rep.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., and Zweig, G. (2016). Achieving human parity in conversational speech recognition. Tech. rep., arXiv:1610.05256.

Yang, S., Luo, P., Loy, C.-C., and Tang, X. (2016). Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5525–5533.

Ye, Q. and Doermann, D. (2015). Text detection and recognition in imagery: A survey. *IEEE transactions on pattern analysis and machine intelligence* **37**, 7, 1480–1500.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters* **23**, 10, 1499–1503.

Zhu, Y., Yao, C., and Bai, X. (2016). Scene text detection and recognition: Recent advances and future trends. *Frontiers of Computer Science* **10**, 1, 19–36.

A Supplementary Appendix

A.1 Description of Matching Procedure

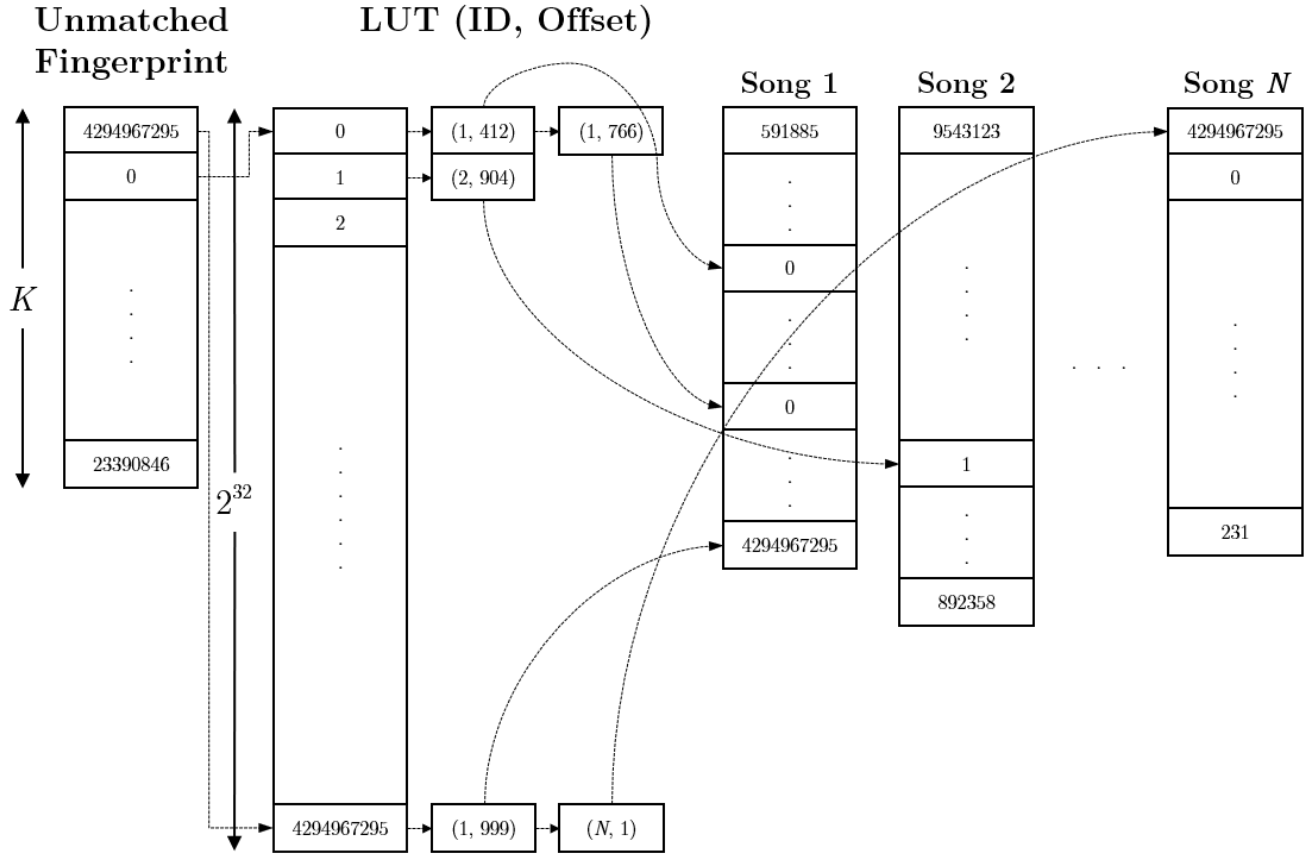


Figure A.1: Diagram for fingerprint database structure and how candidates are chosen for the unmatched fingerprint. The small rectangles containing integers correspond to sub-fingerprints, while the small rectangles containing the (Song ID, Offset) tuples correspond to the values stored in the database. This figure is a modified version of Figure 6 in (Haitsma and Kalker, 2002).

Figure A.1 shows the layout for the fingerprint database we use for matching. We will refer to this figure throughout this section as we explain the procedure for matching.

Denote F_{unk} as the fingerprint we are trying to match, recalling that a fingerprint is an K-dimensional array of 32-bit unsigned integers, where K is determined by the length of the unmatched audio clip.[58] Also define $F_{\text{unk}}(i)$ as the ith sub-fingerprint, depicted by the small rectangles containing integers in Figure A.1. Finding the corresponding match to F_{unk} is an iterative procedure. For each sub-fingerprint $F_{\text{unk}}(i)$, we perform the following steps:

1. Generate a list of candidate matches using the current sub-fingerprint $F_{\text{unk}}(i)$. We extract the list of C tuples $\{(\text{SongID}_j, \text{Offset}_j)\}_{j=1}^{C}$ corresponding to $F_{\text{unk}}(i)$ in the LUT. Each tuple indicates that $F_{\text{unk}}(i)$ occurred in SongID at Offset_j. For example, $F_{\text{unk}}(2)$ occurred in Song

[58] For a 12-second clip, $K \approx 1035$.

1 at offsets 412 and 766, as indicated by the arrows in Figure A.1.

2. Given candidate j, obtain the fingerprint for SongID_j and truncate it to length K in way so that $F_{\text{unk}}(i)$ occurs in the same position of the truncated fingerprint as it does in F_{unk}. We see that $F_{\text{unk}}(2) = 0$ occurs in the second position of F_{unk}, while it occurs in the 412th and 766th position of Song 1. In order to compare the shorter K length unknown fingerprint to Song 1, we need to truncate and align the fingerprint. Denote the truncated and aligned fingerprint as F_j^{trunc}.

3. Compute the bit error rate (BER) between F_{unk} and F_j^{trunc}, where the BER is the the average number of binary digits that differ between the two fingerprints.

4. Declare a match if $BER < 0.3$. Otherwise repeat steps 2–4.

We repeat steps 1–4 for each sub-fingerprint in F_{unk} until either a match is found or until we've checked all sub-fingerprints, at which point we declare no match.[59]

[59]To control for the possibility of errors in computing the fingerprint of the unmatched video, we follow the suggestion of Haitsma and Kalker (2002) and search over sub-fingerprints generated by flipping the three-most unreliable bits. Here, unreliable bits refer to the three bits for which the energy differences given in equation (1) are nearest the decision boundary of 0.

A.2 Additional Coverage Analysis

This appendix presents additional coverage analyses. Figure A.2 shows the histograms for the proportion of videos found on YouTube for each candidate by partisanship whereas Figures A.3 and A.4 presents the same histograms by incumbency status and outcomes of elections, respectively. We find that the differences are not stark, although Democratic candidates tend to have a somewhat higher proportion of matched videos than their Republican counterpart. In addition, we find only small differences by incumbency status and election outcomes though non-incumbents and election winners are likely to have a greater proportion of videos matched than incumbents and losers, respectively.

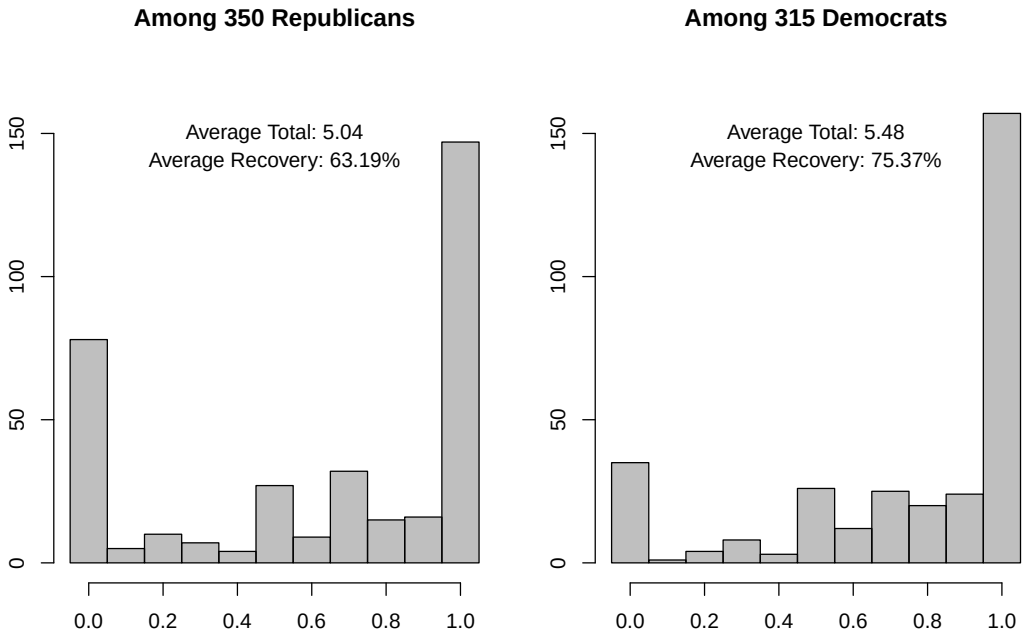


Figure A.2: Proportion of Videos Found on YouTube for Each Candidate by Partisanship. Each histogram represents the distribution across candidates with official YouTube channels for the number of matched videos from YouTube divided by the total number of CMAG videos in the WMP data set. Average Total is the mean number of videos per candidate recorded by the WMP. Average Recovery is the arithmetic (unweighted) mean of proportions of videos found for individual candidates.

We also conduct a regression analysis. Table A.1 reports the results from a Binomial logistic regression predicting whether a YouTube channel was found (the first two columns) and the proportion of matched videos (the remaining columns) using a variety of candidate and election characteristics. Odd-numbered models include independent or third-party candidates as the baseline, while even-numbered models exclude such candidates from analyses and have Democrats as the baseline. For models (3)–(4), we treat the candidates whose channel could not be found as having 0 matched videos, while for models (5)–(6), we exclude any candidate for whom we could not find an official YouTube channel. The interpretation of these results is given at the end of Section 2.3.

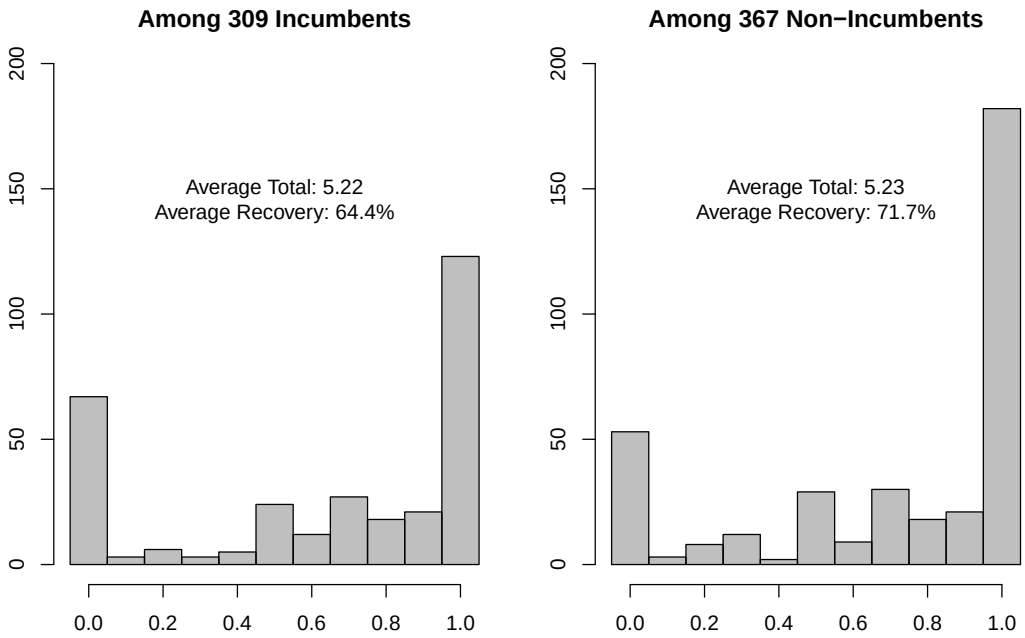


Figure A.3: Proportion of Videos Found on YouTube for Each Candidate by Incumbency Status. See the caption of Figure A.2 for details.

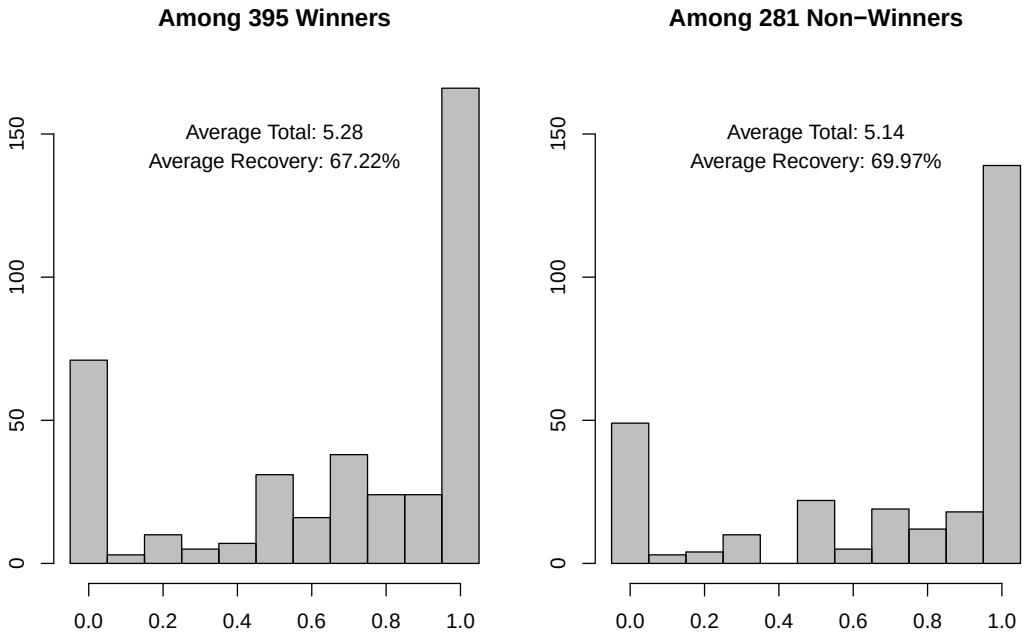


Figure A.4: Proportion of Videos Found on YouTube for Each Candidate by Election Outcomes. See the caption of Figure A.2 for details.

	Dependent variable:					
	Channel Found		Matches Found (All)		Matches Found (If Channel Found)	
	(1)	(2)	(3)	(4)	(5)	(6)
Republican	2.023***	−0.119	1.468***	−0.416***	0.193	−0.416***
	(0.387)	(0.186)	(0.405)	(0.113)	(0.533)	(0.119)
Democrat	2.155***		1.891***		0.611	
	(0.383)		(0.405)		(0.534)	
Incumbent	0.583***	0.616***	0.028	0.044	−0.211	−0.205
	(0.225)	(0.224)	(0.122)	(0.123)	(0.127)	(0.127)
Won Election	0.387	0.325	0.220	0.169	−0.088	−0.111
	(0.211)	(0.212)	(0.121)	(0.123)	(0.129)	(0.130)
House	0.229	0.145	0.192	0.196	0.006	0.027
	(0.256)	(0.270)	(0.147)	(0.149)	(0.165)	(0.166)
Senate	0.150	0.096	0.548***	0.526***	0.135	0.123
	(0.313)	(0.331)	(0.160)	(0.163)	(0.175)	(0.176)
Year 2014	−0.431**	−0.341*	0.275**	0.274**	0.448***	0.431***
	(0.182)	(0.186)	(0.116)	(0.117)	(0.123)	(0.123)
Intercept	−1.010**	1.174***	−1.864***	0.049	0.273	0.897***
	(0.418)	(0.303)	(0.416)	(0.175)	(0.548)	(0.193)
Observations	856	813	856	813	676	665

Note: **p<0.05; ***p<0.01

Table A.1: Binomial Logistic Regression for Predicting Channel and Matches Found Using Candidate and Election Characteristics. The first two models predict the indicator variable for YouTube channels being found, first for all candidates (1) and for Republican/Democratic candidates only (2). The following two models predict the proportion of matched videos found, while treating the candidates without YouTube channels as having zero matched videos. The final two models are the same as the other ones, except that we exclude from the regression analysis candidates whose channels were not found.

A.3 Description of Audio Features

This section provides a more formal description of the audio features we used, which were taken from (Ren *et al.*, 2015). Let $S \in \mathbb{R}^{N \times K}$ denote the spectrogram for an audio signal, with $S_{i,k}$ corresponding to the strength of frequency component k in frame i of the spectrogram.

A.3.1 Short-term Features

Statistical Spectrum Descriptor (SSD): Define the frequency associated with bin k as f_k.[60] The statistical spectrum descriptor for frame i is a combination of the spectral centroid, skewness, kurtosis, flux, and rolloff, which are defined as

$$\text{centroid}_i = \frac{\sum_{k=1}^{K} f_k S_{i,k}}{\sum_{k=1}^{K} S_{i,k}}, \qquad \text{skewness}_i = \frac{m_3(S_i)}{\widehat{\sigma}_{S_i}^3}, \qquad \text{kurtosis}_i = \frac{m_4(S_i)}{\widehat{\sigma}_{S_i}^4}$$

and

$$\text{flux}_i = \|S_i - S_{i-1}\|^2, \qquad \text{rolloff}_i = f_{k^*}, \text{ such that } \sum_{k=1}^{k^*} S_{i,k}^2 = 0.85 \sum_{k=1}^{K} S_{i,k}^2$$

where $m_3(S_i)$ and $m_4(S_i)$ are the sample third and fourth central moment of the spectrum, respectively, and $\widehat{\sigma}_{S_i}$ is the sample standard deviation.

Mel-frequency Cepstral Coefficients (MFCC): The exact computation of the MFCC is beyond the scope of the paper, instead we defer the reader to (Lee *et al.*, 2009) for a more formal description and give a high level overview here. The MFCC computes a low-dimensional representation of the pooled log energies contained in a set of logarithmically-spaced subbands designed to mimic the human auditory system[61] using a Fourier-like transform called the discrete cosine transform. This procedure aims to give a low-dimensional representation of the spectral envelope of the spectrum. We retain the first 20 coefficients of the MFCC and use the energy contained in the entire frame to form the feature vector.

Octave Spectral Contrast (OSC): The OSC characterizes differences in the peaks and valleys of the spectrum across 8 logarithmically-spaced subbands. Formally, for each frequency band a, we sort the bins in the subband in order of increasing magnitude. Defining the number of bins in band a as N_a and $P_{a,k}^i$ as the magnitude for the kth bin in band a in frame i, we take the smallest and largest 20% bins and compute the peak and valley as

$$Peak_i(a) = \log\left(\frac{1}{\lceil 0.2N_a \rceil} \sum_{k=1}^{\lceil 0.2N_a \rceil} P_{a,k}\right), \qquad Valley_i(a) = \log\left(\frac{1}{\lceil 0.2N_a \rceil} \sum_{k=1}^{\lceil 0.2N_a \rceil} P_{a,N_a-k+1}\right).$$

Taking the difference between the peaks and valleys yields the contrast for band a in frame i. The OSC feature matrix is formed by concatenating the contrast and the values together.

Spectral Flatness Measure/Spectral Crest Measure (SFM/SCM): SFM/SCM quantify how noise-like or how tone-like the audio signal is. Like the OSC, these quantities are computed in

[60] Converting from bin k to frequency f_k is done via $f_k = k \cdot F_s / N_{FFT}$, where $F_s = 22050$ and $N_{FFT} = 1024$.

[61] For example, a human would perceive a 500Hz and 1000Hz tone to be as different as 2000Hz and 3000Hz tone. The perceptual spacing between tones is nonlinear on the Hz scale.

8 logarithmically-spaced subbands. Using the same definitions in the OSC section, the SFM/SCM is computed as

$$SFM_i(a) = \frac{\sqrt[N_a]{\prod_{k=1}^{N_a} P_{a,k}}}{\frac{1}{N_a}\sum_{k=1}^{N_a} P_{a,k}}, \qquad SCM_i(a) = \frac{\max_{k=1,\ldots,N_a} P_{a,k}}{\frac{1}{N_a}\sum_{i=k}^{N_a} P_{a,k}}.$$

An SFM of 1 implies the spectrum has relatively equal magnitudes at all frequencies, which corresponds to audio that sounds like white noise, such as radio static. Low spectral flatness suggests that the energy in the spectrum is concentrated around only a few frequency bands, corresponding to an audio signal that sounds like a mixture of tones. An SCM of 1 corresponds to a flat, noise-like spectrum, while and SCM much larger than 1 indicates the spectrum is "spiky".

A.3.2 Long-term Features

Modulation Feature Spectrogram The modulation feature spectrogram is used to compute the long-term features, which aim to capture information regarding the rhythm, tempo, and beat of an audio signal. They are constructed from the short-term feature matrices. Formally, given a short-term feature matrix $F \in \mathbb{R}^{N\times D}$ and defining the dth column F as F_d, we compute the modulation spectrogram via the following steps:

1. For each feature d, compute the spectrogram corresponding to the feature signal F_d using a segment length $W = 256$ and an overlap factor of $1/2$. This step produces a $T \times 129$ matrix M_d, where T depends on the number of frames N in the original spectrogram.

2. Collapse the matrix M_d to a single vector $m_d \in \mathbb{R}^{129}$ by taking the average over all rows. m_d characterizes the frequency content of feature d.

3. Row stack the vectors m_d to form the modulation feature spectrogram $M_F \in \mathbb{R}^{D\times 129}$.

As was the case for the short-term feature, the long-term feature is a characterization of the modulation feature spectrogram, and e use the septal-based spectral contrast as th. For a modulation feature spectrogram M_F, we partition this matrix along the columns into 7 logarithmically spaced subbands. Defining the number of bins in band a as N_a and $P^d_{a,k}$ as the magnitude for the kth modulation frequency bin in band a for feature dimension d, the valleys and peaks in are computed as

$$MPeak(d,a) = \max_{k\in a} P^d_{a,k} \qquad MValley(d,a) = \min_{k\in a} P^d_{a,k},$$

We form the contrast by taking the difference between the peaks and valleys, all of which are $D \times 7$ matrices. We take the mean and standard deviation along the rows and columns of the valley and contrast matrices, separately, and stack these together to produce a feature $f \in \mathbb{R}^{4D\times 28}$. We independently apply this procedure to the MFCC feature matrix, the OSC feature matrix, and the SFM/SCM feature matrix and concatenate these all together to form the long-term feature $LT \in \mathbb{R}^{296}$.

Joint-Frequency Feature The averaging step in computing the modulation feature spectrogram throws out some information regarding the temporal evolution of the modulation features. The purpose of the joint-frequency feature is to recover and characterize this lost information. After obtaining the spectrogram $S \in \mathbb{R}^{N\times 513}$ described at the beginning this Section 3.2.3, another

one-sided FFT is performed along each column of S, followed by an absolute value operation, yielding a matrix $J \in \mathbb{R}^{N/2+1 \times 513}$. This matrix is then partitioned into a 7×8 grid using the same logarithmically-spaced subbands as the octal- and septal-based spectral contrast features described above. Each block of the grid is vectorized, a we compute the spectral contrast, spectral valley, spectral flatness measure, and spectral crest measure as defined in the short-term feature section. This process produces four matrices, each of dimension 7×8. We vectorize and stack the matrices together to form the joint-frequency feature $JF \in \mathbb{R}^{224}$.

Combining all of the features described above forms our 636-dimensional audio feature.

A.4 A Comprehensive List of Keywords Used for Issue Detection

Issue/Figure	Keywords	Frequency in WMP
Barack Obama	obama|the president|our president	603
George W. Bush	bush	21
Ronald Reagan	reagan	5
John Boehner	boehner	7
Nancy Pelosi	pelosi	52
Mitch McConnell	mcconnell	26
Harry Reid	reid	14
Congress	congress NOT in congress|for congress|to congress †	419
Democrats	democrats|and democrat|or democrat	229
Republicans	republicans|and republican|or republican|GOP	270
Tea Party	tea party	43
God	god NOT thank god †	8
Hope	hope NOT i hope|we hope †	21
Change	change	121
Experience	experience	50
Liberal	liberal	76
Conservative	conservative	163
Special Interest	special interest	71
Negative Campaigning	negative campaign	18
Main Street	main street	6
Wall Street	wall street NOT wall street journal †	87
Big Government	big government	23

Table A.2: Comprehensive List of Keywords Used for Automated Detection of Issues/Figures, Cont'd (Political Figures and Actors/Words and Phrases).
† Exclusion rules are applied.

Tables A.2 to A.4 show the list of keywords associated with each topic arranged in the order they are defined by the WMP and grouped together in broad categories.

Issue/Figure	Keywords	Frequency in WMP
Tax	tax	769
Deficit/Budget/Debt	deficit|budget|debt	482
Government Spending	spending	309
Recession/Stimulus/Bailout	recession|stimulus|bailout	86
Minimum Wage	minimum wage	51
Farming	farm|agricult	62
Business	business	351
Union	union	11
Jobs/Employment	jobs|outsourc|employment|unemploy| out of job|back to work + many forms of [create job], [lose job], [kill job]	1105
Poverty	poverty	14
Trade/Globalization	trade|globaliz|NAFTA NOT cap and trade †	113
Housing/Subprime Mortgage	housing|subprime|sub prime	38
Economy (Generic)	economy|economic prosperity	273
Inequality	unequal|inequal|equal pay	53
Abortion/Women's Health	abortion|pregnan|woman's right to choose| women's right to choose|reproductive right| pro choice|pro life|woman's health| women's health|birth control|contracept| planned parenthood + various forms of [Roe v. Wade]	213
Homosexuality/Gay/Lesbian	lgbt|gay|lesbian|transgender|same sex| marriage equality|traditional marriage| one man and one woman	16
Moral/Family/Religious Values	honesty|integrity|moral|family value	100
Tobacco	tobacco|cigarette	0
Affirmative Action	affirmative action	2
Gambling	gambling	0
Assisted Suicide/Euthanasia	euthanasia|assisted suicide	0
Gun Control	gun|second amendment|2nd amendment| bear arms|NRA	63
Civil Liberty/Privacy	civil libert|freedom of speech|free speech| freedom of religion|freedom of faith|privacy	12
Race Relations/Civil Rights	civil right	6
Crime	crime|criminals|violence|victim|predator	47
Narcotics/Drugs	drugs|drug addict|drug deal|drug lord|narcotic| marijuana|cocaine|heroin|opioid|opiate NOT prescription †	11
Capital Punishment/Death Penalty	death penalty|capital punishment	2
Supreme Court/Judiciary	judicia|supreme court|courts	4

Table A.3: Comprehensive List of Keywords Used for Automated Detection of Issues/Figures, Cont'd (Economy/Society/Law and Order).
† Exclusion rules are applied.

Issue/Figure	Keywords	Frequency in WMP
Education/Schools	educat|schools|tuition|affordable college| college affordable|college more affordable	445
Lottery for Education	lottery for education	4
Child Care	childcare|child care|daycare|day care| care for your child|care for our child| care for your kid|care for our kid	17
Healthcare	healthcare|health care|health insurance| medical insurance|obamacare|affordable care	428
Prescription Drug	prescription drug	22
Medicare	medicare	417
Social Security	social security	254
Welfare	welfare	26
Military	military|troops|armed force|servicemember| service member|in uniform|war in|wars in	145
Foreign Policy	foreign policy	19
Veterans	veteran|vet|VA|world war|vietnam war	244
Foreign Aid	foreign aid	4
Nuclear Proliferation	nuclear proliferation|nuclear weapon	3
China	china|chinese	68
Middle East	middle east	16
Afghanistan	afghan	26
September Eleven	911|september eleven|nine eleven|september 11	7
Terror/Terrorism/Terrorist	terror|war on terror	35
Iraq	iraq	54
Israel	israel	2
Iran	iran	4
Environment	environ|EPA|cap and trade	32
Global Warming	global warming|climate change	12
Energy	energy|power plant|keystone|pipeline|coal| petroleum|natural gas|solar|fracking	195
BP Oil Spill	oil spill	4
Campaign Finance Reform	campaign finance|citizens united	13
Government Ethics/Scandal	corrupt|government ethics|government scandal	144
Corporate Fraud	corporate fraud	22
Term Limit	term limit	6
Pledge of Allegiance	pledge of allegiance|pledge allegiance	0
Immigration	immigra|alien|border|dream act|amnesty	66
Local Issues	local	186
Government Regulation	regulation	118

Table A.4: Comprehensive List of Keywords Used for Automated Detection of Issues/Figures, Cont'd. (Welfare/Defense and Foreign Relations/Environment/Other).
† Exclusion rules are applied.

A.5 Scripts for Amazon Mechanical Turk Classification Tasks

*****IMPORTANT*** READ THE INSTRUCTIONS BELOW CAREFULLY.** (Click to collapse)

Responses that do not follow the instructions will not be eligible for payment.

1. Watch the following video and answer simple questions: (i) whether a given topic was mentioned/pictured in the ad, and (ii) if so, how. Some topics may be followed by special instructions.

2. You can only complete up to 50 HITs (shared between our 15-, 30-, and 60-second versions). Accepting more than 50 will display an error message for you so that you don't accidentally have your extra submissions rejected.

3. Our payment-per-HIT allows ample time to watch each video at least twice in full so as to ensure maximum accuracy. If you are unsure about how to answer, please go back to the video and re-watch. We already have established benchmarks for accuracy from a previous study and unfortunately cannot grant payment to work that falls significantly below reasonable quality (e.g., random guesses, snap decisions).

4. Below are some general guidelines for answering:

- We define "mentioned" as either (i) verbally spoken about or (ii) appering as on-screen text. For example, if an on-screen text reads "Congress is broken," then this should count as a mention of **Congress**.
- The topics need NOT be mentioned as a main theme to be counted. For example, "as a self-accomplished businesswoman, I know what it takes to create jobs." should still qualify as a mention of the subject **business**, even though the keyword was only brought up in passing.
- The topic keyword can take different forms. For example, "a liberal," "liberals," and "liberal politicians" should all count as mentions of the topic **liberal**.
- Some topics may have different wordings or natural extensions. For example, "assisted suicide" is just a different wording of the subject **euthanasia** and thus should be counted as mentioning it. Also, **gun control** should include any natural extensions such as "Second Amendment" or "right to keep and bear arms."

In the following video, we are interested in whether it mentions or pictures **Immigration**.

If the video does not appear above, click here to access our backup video.

1. Does the video mention or picture Immigration?

- ○ Yes
- ○ No

2. If you answered "Yes" above, why? (Choose ANY/ALL that apply)

- ☐ The topic was verbally mentioned.
- ☐ The topic appeared as on-screen text.
- ☐ A picture of the topic was shown.

Figure A.5: An Example Script for Issue Detection Task Exactly as Seen by Amazon MTurkers.

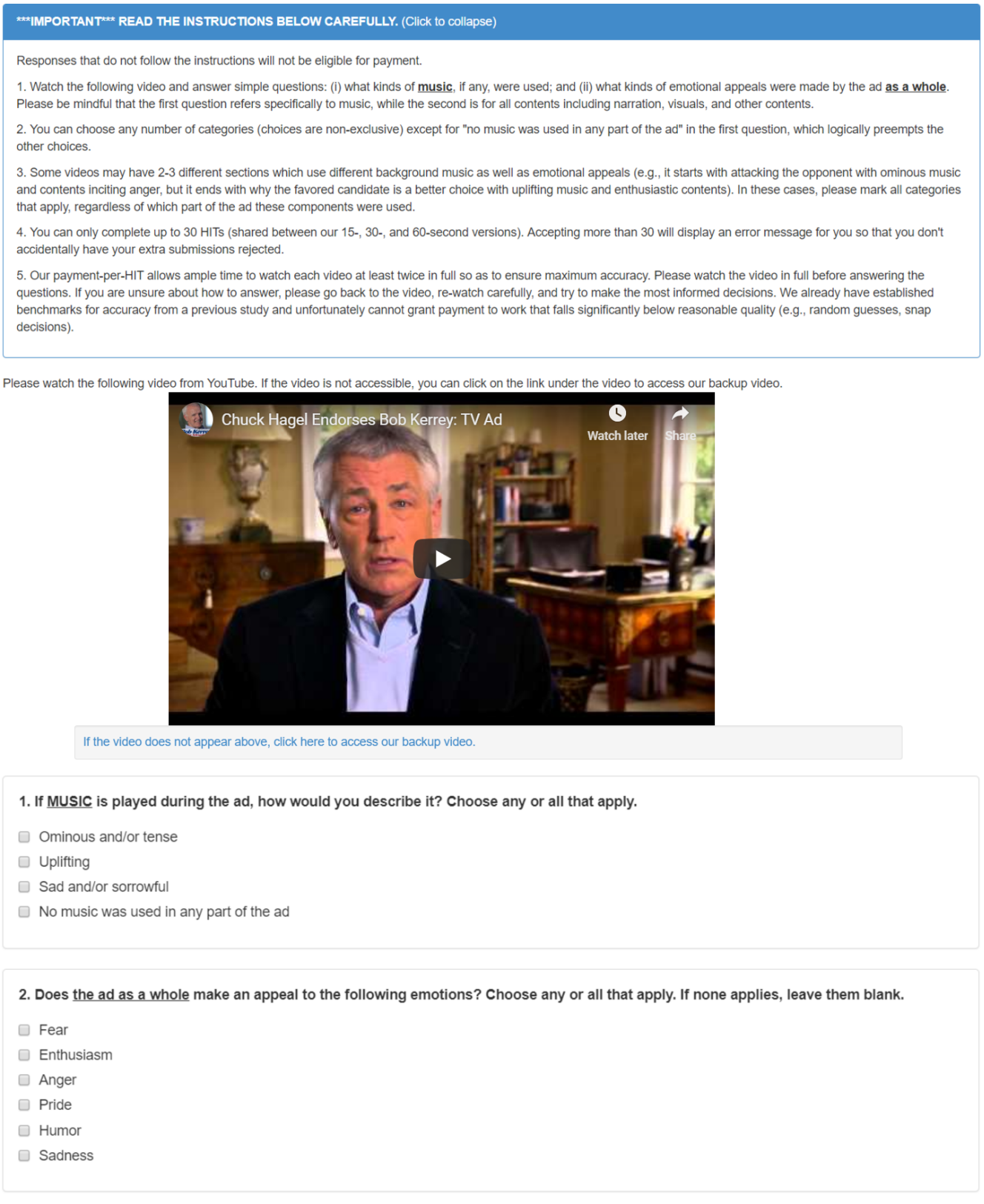


Figure A.6: An Example Script for Musical Mood Detection Task Exactly as Seen by Amazon MTurkers. The script also includes a second question on the emotional appeal of the ad, which was not used in the paper.

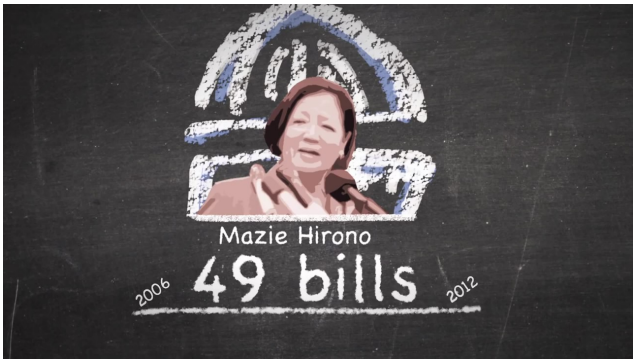
A.6 Additional Examples

(a) Ad by a House candidate Patrick Henry Hays (Democrat, Arkansas). The automated coding algorithm detected texts written in the poster in the background (though it is difficult to see in this picture), the first word of which coincided with the last name of his opponent, French Hill.

(b) Ad by a House candidate Alex Mooney, (Republican, West Virginia). The automated coding failed to correctly read the handwritten name of the opponent: it read 'Nick Case,' thus omitting 'y' which lead to incorrectly returning No.

Figure A.7: Examples Illustrating the Mistake of the Automated Coding for the Issue Mention Variable.

(a) Ad by Senator Jeff Flake (Republican, Arizona). The automated coding algorithm did not detect the angled, sideways picture of his opponent, Rich Carmona (leftmost figure), leading to a false negative.



(b) Ad by a Senate candidate Linda Lingle (Republican, Hawaii). The image of her opponent, Mazie Hirono, was heavily processed so that the automated coding algorithm did not detect it, leading to a false negative.

Figure A.8: Examples Illustrating the Mistake of the Automated Coding for the Candidate Appearance Variable.

A.7 Additional Validation Results

		Majority Opinion among MTurkers					
		Ominous/Tense		Uplifting		Sad/Sorrowful	
		No	Yes	No	Yes	No	Yes
WMP coding	No	271 (61.59%)	30 (6.82%)	113 (25.68%)	18 (4.09%)	348 (79.09%)	31 (7.05%)
	Yes	32 (7.27%)	107 (24.32%)	39 (8.86%)	270 (61.36%)	27 (6.14%)	34 (7.73%)

Table A.5: Comparison of the Musical Mood Variables between the WMP and MTurker Codings. MTurk coder responses are transformed to a binary variable based on the majority opinion. The value in each cell corresponds to the frequency of the four different combinations of results from the WMP and automated coding schemes. The three two-by-two matrices correspond to the three different moods of music used in the WMP data. The results shown here are from the test data set of size 440.

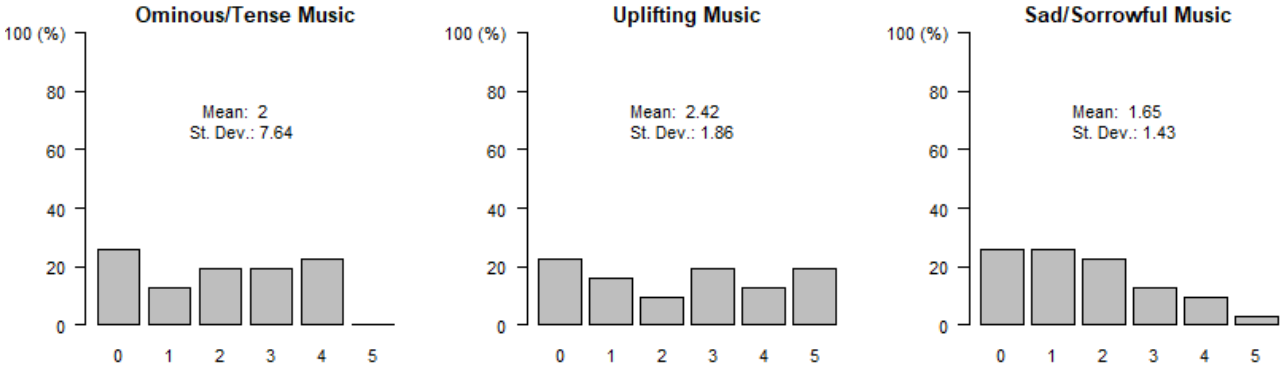


Figure A.9: Number of MTurkers Who Code the Music into Three Categories among the 31 Cases of Disagreements between Two WMP Coders. This shows that the mood of music in these videos is inherently difficult to code.

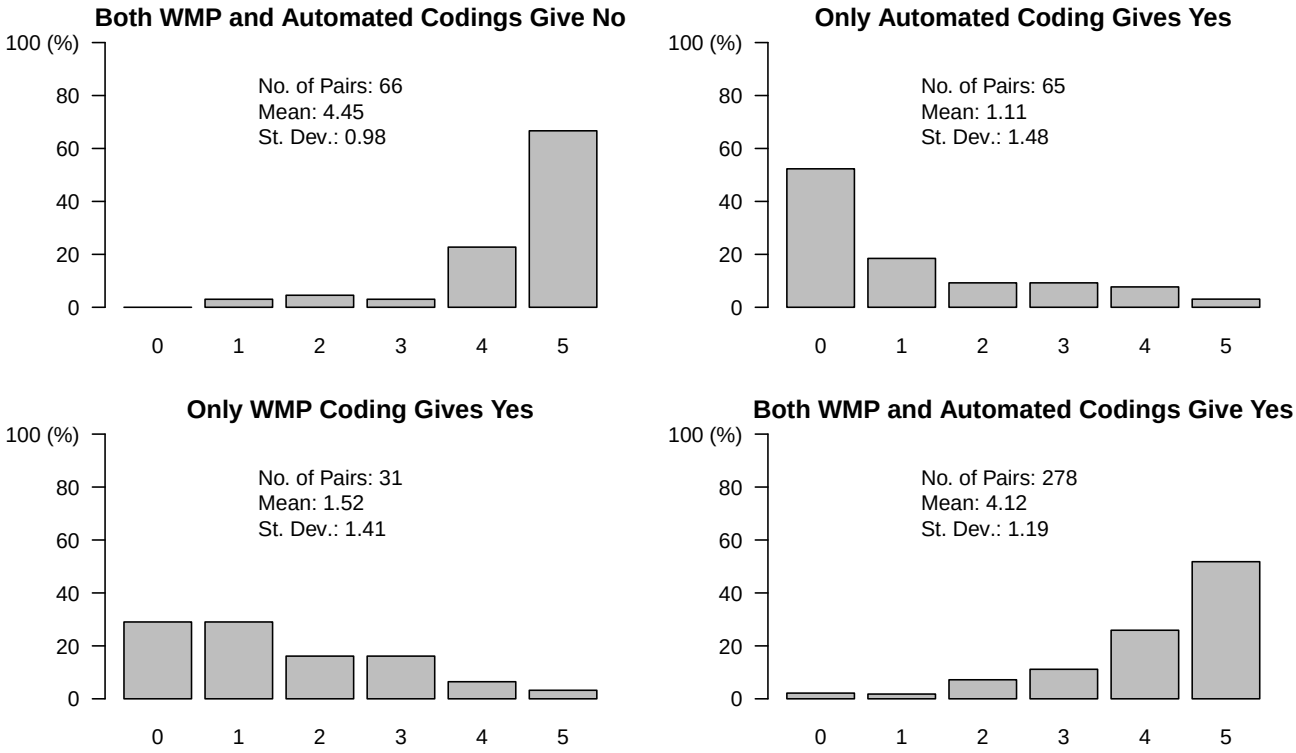


Figure A.10: Number of MTurkers Who are in Agreement with Automated Coding For Uplifting Music. Four cases are based on the agreement and lack thereof between the WMP and automated codings. A total number of MTurkers for each task (hence the maximum possible number of MTurkers who agree with automated coding) is five. The texts within each plot show the number of issue-video pairs included in each sample as well as the mean and standard deviation of the number of MTurkers in agreement with the automated coding.

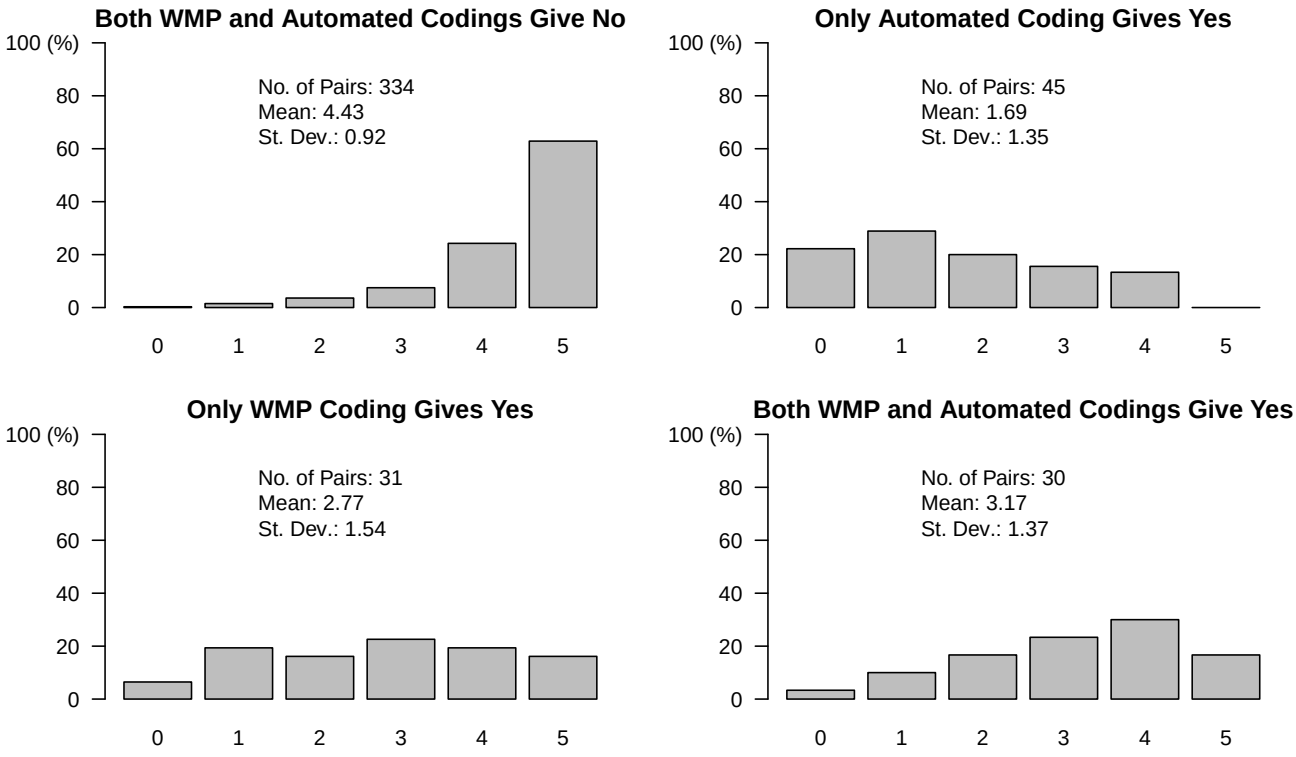


Figure A.11: Number of MTurkers Who are in Agreement with Automated Coding For Sad/Sorrowful Music. See the caption of Figure A.10 for details.

		Naive Bayes coding					
		Text Only		Music Only		Text and Music	
		Negative	Positive	Negative	Positive	Negative	Positive
WMP coding	Negative	313 (60.42%)	12 (2.32%)	236 (45.56%)	89 (17.18%)	262 (50.58%)	63 (12.16%)
	Positive	61 (11.78%)	132 (25.48%)	82 (15.83%)	111 (21.43%)	72 (13.90%)	121 (23.36%)

Table A.6: Comparison of the Ad Negativity Variable between the WMP and Automated Codings using the Naive Bayes Classifier. The results on the top are from automated tone classifiers using non-linear SVM with radial basis function, and those on the bottom are from using Naive Bayes classifier. The value in each cell corresponds to the frequency of the four different combinations of results from the WMP and automated coding schemes. The three two-by-two matrices correspond to the three types of input data used to train the models.