Unsupervised Speaker Identification For TV News

Daniel N. Woo and Ramazan S. Aygun, *Member*, *IEEE*

D. N. Woo is a Master's graduate from the University of Alabama, Huntsville, AL 35899 USA (email: dnw0012@alumni.uah.edu).

R. S. Aygun is an Associate Professor in the Computer Science Department, University of Alabama, Huntsville, AL 35899 USA (email: raygun@cs.uah.edu).

***Abstract*—Identifying the speakers in TV news would help to analyze and understand the news content. Previous research has identified speakers on pre-trained faces for TV shows and movies. News videos are challenging because new faces often appear. By using an unsupervised method, this paper proposes to label speakers using just the available information in the news video without external information. Our proposed framework segments the audio by speaker, parses closed captions for speaker names, identifies talking persons, and performs optical character recognition for speaker names. Our framework utilizes face recognition, face clustering, face landmarking, natural language processing tools, and speaker diarization. Our results indicate 63.6% accuracy for identifying speakers for CNN news.**

***Index Terms*—Television, Unsupervised learning, Face recognition, Face clustering, Face landmarking, Natural language processing, Speaker diarization, Closed captions**

I. Introduction

Television (TV) networks produce a tremendous amount of information every day. Identifying who is speaking at an instant may help structural organization of the news content, and we may answer questions such as “what did *person X* say about an issue?” In the past, people recognition has been studied by first training a system using a set of faces and audio with labeled names.

In this paper, we propose a method to identify who is speaking throughout a video without providing external information or training. We search for possible speaker names within the three components of TV news: video, audio and closed captions. Firstly, our system checks whether the name of a speaker appears on screen while the person speaks. Secondly, our system determines the most common name to appear on screen during a speaker’s audio segments. Thirdly, our system parses the closed captions to see if the name of a speaker is used. By using weighted majority voting to combine this information, a text transcript with speaker names can be produced.

As a proof of concept, we focus on CNN news because it is more consistent than other networks in introducing speakers and displaying the speaker's name on screen. One of the goals of our system is to label speakers in closed captions since speaker transitions are marked with a ">>" prompt, but no speaker name is displayed except for prerecorded TV shows. The underlined names of speakers (which are not part of the script) can be added to a sample of closed captions from CNN Newsroom, May 5, 2013, 12:00 pm as follows:

>> **WOLF BLITZER:** ARE THEY SAYING ANYTHING ABOUT HOW THE FIRE MAY HAVE STARTED AT LEAST SO FAR, DAN?

>> **DAN SIMON:** NOT YET, WOLF. OTHER THAN TO SAY THAT IT LOOKS LIKE IT STARTED IN THE BACK OF THE VEHICLE.

In this paper, our goal is not to improve stand-alone existing algorithms such as face recognition. Our system rather utilizes the existing technologies and builds a framework to automatically label the speakers. This is a challenging research problem since speaker labeling is based on the information available in the news broadcast. Our method can also be inspiring for other application domains where people need to be recognized without training. Since our system links text to people, further data analysis or information retrieval techniques can be used to support multimedia search engines.

II. Related Work

Face detection is the first step in recognizing who is speaking. The Viola-Jones algorithm is commonly used to search an image for faces.[1] By training image detectors on a set of images and using an AdaBoost learning algorithm, faces are detected. The Luxand FaceSDK, which is a commercial library, can perform face detection, recognition, clustering, and landmarking.[2] After faces are detected, the next step is to *recognize faces*. Eigenfaces is a principal components analysis (PCA) technique in which a training set of images is separated into Eigenfaces.[3] Fisherfaces is a linear discriminant analysis (LDA) technique, which attempts to separate a trained set of faces using classes.[4] Both techniques require a training step with pre-labeled faces, so they are not appropriate for recognizing faces in news videos where unknown faces may appear throughout the video.

Since the same face may appear many times in a video, it is possible to cluster faces of the same person. Local binary patterns (LBP) are a texture-based approach to recognizing similar objects.[5] It can generate a similarity value between 0 and 1 using the normalized X^2 difference. It can be used to identify similar faces[6] and cluster unknown faces without training.

To determine who actually is speaking in a video requires *face landmarking*, which determines the important points on a face, like the upper point of the mouth and the lower point of the mouth to identify talking faces. Active shape models (ASM) tries to match a model of the face onto an image's edges.[7] It is an iterative algorithm that minimizes least squares error. An open-source face landmarking library is now available.[8]

Text information in the video frames must be captured to see if names of people appear in video frames. After converting the video frames to grayscale, the image is converted to black and white by sliding a window over the image using an adaptive thresholding technique.[9] There are many reliable OCR (optical character recognition) algorithms such as the Tesseract OCR.[10]

After video frames have been converted to text, the Stanford CoreNLP is used to identify when proper names occur in text, using its parts-of-speech identification and named entity recognition (NER).[11] The CoreNLP inputs text and outputs a XML file where every word has its parts-of-speech identification and named entity recognition, if found.

Speaker diarization identifies whenever the same person speaks throughout an audio stream by outputting time segments with a speaker cluster number. The LIUM_SpkDiarization is used for speaker diarization.[12] Names of faces are determined in entertainment TV shows and movies for a limited known cast using face recognition trained on pre-labeled faces.[13,14,15]

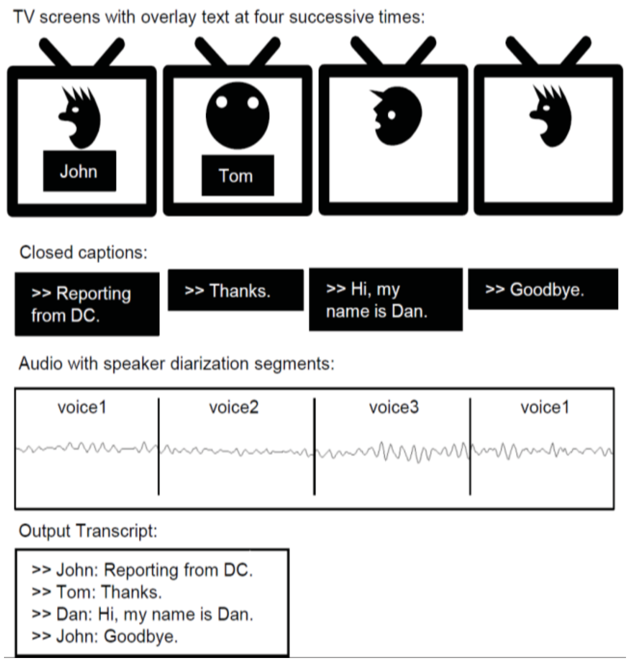
The proposed systems by Stein et al.[16,17] aim to recognize people who appear frequently. 2-minute speeches are used for training for identifying 253 politicians.[16] They achieved 8.06 Equal Error Rate (EER) using GMM with 1024 mixtures. Identified face clusters were labeled by a human annotator. Two-thirds of data were used for training for speaker identification (with 10% EER for 32 speakers).[17] They assumed only one person is visible when a banner is visible and did not check who actually was speaking by following mouth movements.

III. Speaker Identification

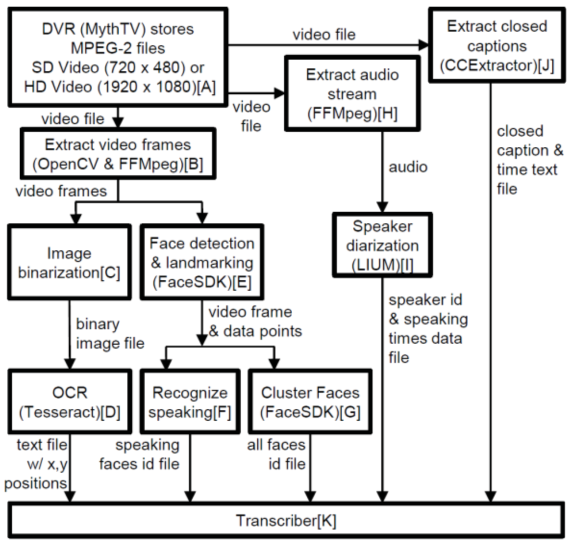
In this section, we explain the framework and modules of our system. Fig. 1a shows the three ways to find a speaker name and how a speaker name is copied to audio segments that have the same voice. In the first frame, the name of the speaker appears at the bottom, and this can be used to label the speaker. However, when this speaker talks again in the last frame in Fig. 1a, the name of the speaker does not appear again. We can use speaker diarization or face matching to determine if they are the same people. According to the speaker diarization, the first speaker and the last speaker are the same; so we can label the last speaker also as ‘John’. There is no text appearing on screen for the third speaker; however, his name can be identified from the closed captions. Finally, names from audio/video segments are added to the closed caption text to produce an output transcript.

A. *The Framework for Speaker Identification*

Fig. 1b depicts the data flow diagram of our framework. Many of the functional blocks use open-source third-party programs and libraries. Two blocks use the Luxand FaceSDK to detect, recognize, landmark, and cluster faces. The names of the programs appear in parentheses.



(a)


DVR (MythTV) stores MPEG-2 files SD Video (720 x 480) or HD Video (1920 x 1080)[A]
video file
Extract closed captions (CCExtractor)[J]
Extract audio stream (FFMpeg)[H]
video file
video file
Extract video frames (OpenCV & FFMpeg)[B]
video frames
audio
closed caption & time text file
Image binarization[C]
Face detection & landmarking (FaceSDK)[E]
Speaker diarization (LIUM)[I]
binary image file
video frame & data points
speaker id & speaking times data file
OCR (Tesseract)[D]
Recognize speaking[F]
Cluster Faces (FaceSDK)[G]
text file w/ x,y positions
speaking faces id file
all faces id file
Transcriber[K]

(b)

Fig. 1. (a) Three ways to find a speaker name and how a speaker name is copied to audio segments with the same voice, and (b) data flow diagram with program and library names

The video files coming out of the MythTV 0.25 DVR (block A in Fig. 1b) represent the three sources of information that will be used to identify the speaker. The video source works on video frames and uses face detection, recognition, landmarking, and clustering to identify talking faces. Then, image binarization and OCR are used to find the names for the faces. The text source uses closed caption data extracted from the video file to find speaker names. The audio source uses the audio stream from the video file for speaker diarization to segment the audio stream by speaker. Each source of information produces a set of output files. All of this information is combined by the transcriber using weighted majority voting to identify the name of the speaker and produce a readable text transcript.

B. Speaker Identification using Face Recognition and OCR

By detecting frontal faces, the purpose is to identify whenever the same talking person appears throughout a video file and to find his or her name. OpenCV 2.4.4 extracts video frames from the MPEG-2 file (block B in Fig.

1b). It uses FFmpeg 1.1.1 to perform the extraction. Video frames are processed by the Luxand FaceSDK 4.0 (block E in Fig. 1b). The critical parameter for Luxand FaceSDK is InternalResizeWidth, which is an indicator of the face region to image size and was used as 256 in our experiments, allowing recognition of faces down to size 57x57 in the original 720x480 image. The library recognizes frontal faces with +/- 30 degrees of rotation. For each face, the library landmarks facial features such as the top of the mouth and the bottom of the mouth. A basic algorithm can determine which face is talking (block F in Fig. 1b) by identifying which mouth is changing vertical size the most. Faces of the same person are clustered together (block G in Fig. 1b).

1) Face Detection and Recognition

The FaceSDK face detection is used to find every frontal face in a video frame. A cluster of similar views of that same person is stored as a *pose*. Since different angle poses of the same individual are not matched by the FaceSDK, each cluster has approximately the same pose angle. These different angle poses must be connected as the same person. For this paper, we use the terminology “connecting poses” as identifying different poses of the same person and linking those poses.

The algorithm for face detection is provided in Algorithm 1. Firstly, faces in the current frame are detected. If the current scene is the continuation of the previous scene, faces in the current frame are connected to faces in the previous frame. Mouths of people are detected next and checked if they move or not for the previous 6 processed frames. Each face is compared with all faces in all poses. The current face is added to the pose of the most similar matching template if the pose has less than 10 face templates. If there is no good match, a new pose with the current face is created.

Let *videoFile* represent a pointer to a video file. Let I_n represent n^{th} frame from *videoFile*. Because video contains a lot of repetitive information, only every 5th frame is processed. Let F_c and F_p present an array of faces for the current image and previous image, respectively. *POSES* maintain all poses of faces in the database. Each face f in F_c stores the image coordinates and landmarking data. The face template includes 16KB feature set used to match similar faces. The landmark data includes the top and bottom image coordinates of the mouth. This is used to find

Digital Object Indentifier 10.1109/MMUL.2016.3 1070-986X/$26.00 2016 IEEE

talking faces. The function IdentifyTalkingFace is called to detect which face is talking and is used to merge results with speaker diarizarion and match the text (OCR) name with the face.

Algorithm 1 FaceDetection(videoFile)

```
// IN: videoFile
// OUT: F, POSES
$n$=0; $F_p$={} // frameNo; list of faces in the previous frame
do
    $I_n$=getNextFrame($videoFile$,$n$);
    $F_c$=findFacesInFrame($I_n$)
    if not sceneChangeDetected($I_n$,$I_{n-5}$) then
        connectFacesWithPreviousFaces($F_c$,$F_p$)
    foreach face $f$ in $F_c$ do
        findLandMarks($f$)
        pose=POSES.findTheMostSimilarPose($f$)
        if (pose.noOfFaces<10) then
            pose.addFace($f$)
        elseif pose.increaseDiversity($f$) then
            pose.swapWithTheMostSimilarFace($f$)
        storeMouthSizesInCircularBuffer($f$)
        identifyTalkingFaces($F_c$)
    endfor
    $n$+=5; $F_p$= $F_c$;
until endOf(videoFile)
```

Digital Object Indentifier 10.1109/MMUL.2016.3 1070-986X/$26.00 2016 IEEE

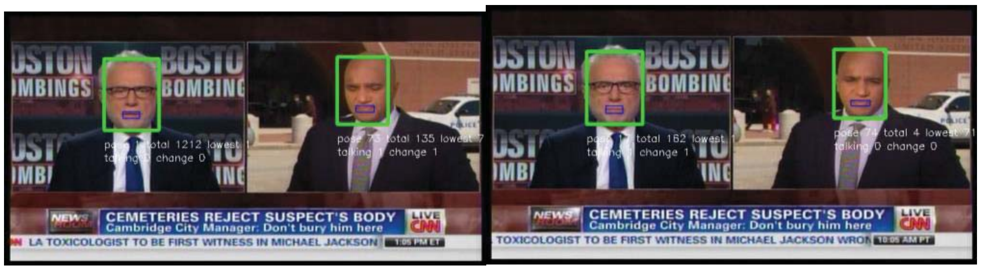
The algorithm to process a face match is provided in Algorithm 2. We need to keep a variety of templates that are different from each other in a pose. Whenever there are 10 templates in a pose and there is a match with the current face template, the current face template may replace the most similar template in the pose to add a variation to the pose.

Algorithm 2 pose.increaseDiversity(f)

```
// IN: pose: the current pose
// IN: f: the possible replacement face
// OUT: boolean
pose.s=pose.findTheMostSimilarFace(f)
oldSimilarity=pose.computeSumOfSimilaritiesToAllFacesInPose(pose.s)
newSimilarity=pose.computeSumOfSimilaritiesToAllFacesInPose(f,pose.s)
    // ignore pose.s in summation
if newSimilarity<oldSimilarity then
  return false;
return true;
```

In Fig. 2, two faces are recognized. A big box surrounds each face. A smaller box surrounds each mouth. The left person is recognized in pose 1, and the right person is recognized in pose 73.

(a)


CEMETERIES REJECT SUSPECT'S BODY
Cambridge City Manager: Don't bury him here
TOXICOLOGIST TO BE FIRST WITNESS IN MICHAEL JACKSON

(b)

Digital Object Indentifier 10.1109/MMUL.2016.3 1070-986X/$26.00 2016 IEEE

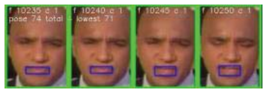
(c) 

Fig. 2. (a) The right person is recognized in pose 73 (left) and 74 (right), (b) faces in pose 73, and (c) faces in pose 74.

2) *Different Angle Poses*

Because different angle poses of the same person in a scene may not be recognized as the same person, these different poses must be connected. A second pass through the generated data is required to connect all the different angle poses of a person. Fig. 2a shows successive video frames from the same file. The right person is recognized in pose 74 in the second frame. This pose must be connected to pose 73 in the first frame, so the poses can be recognized as the same person. Fig. 2b and 2c show the faces in poses 73 and 74, respectively. While the faces in each pose are similar, the faces between poses are slightly different, usually with a slightly different pose angle.

Connecting Different Angle Poses. If a face appears in the same (x,y) position as a previous face and a histogram-based scene change detector has not found a screen change,[18] it is assumed that the face is the same person. If two faces appear at the same position in consecutive frames and their poses are different, this means that the face similarity function does not consider them the same person due to different angle poses but they actually belong to the same person. The different poses are connected by storing two pose numbers in a list: the current pose number and the pose number of the previous face. These two poses are connected to be considered as the same individual and added to a global list of connected pairs of pose numbers, *CP*. Additionally, the vertical mouth size data is copied from the previous pose to the current pose. The lowest pose number is used as the person identifier for all poses that belong to the same person.

3) *Face Landmarking and Speaking Recognition*

Talking faces are identified using face landmarking in the Luxand FaceSDK, which returns the (x,y) position of 66 facial landmarks. Each pose stores the vertical mouth size of the same person in the previous 6 frames. If the new pose belongs to the same person, the old pose's vertical mouth sizes are copied to the new pose. After processing each frame, the face with the most average change in vertical mouth size is identified. This is

Digital Object Indentifier 10.1109/MMUL.2016.3 1070-986X/$26.00 2016 IEEE

determined to be the talking face, and this information is stored in the face array. Fig. 2a shows successive video frames from CNN Newsroom, May 5, 2013, 12:00 pm. Boxes are shown around their faces and mouths. Since the left person’s mouth size changes the most, he is recognized as talking.

4) *Image Binarization and OCR*

Video frames are binarized for OCR (block C in Fig. 1b). For image binarization, the algorithm matches each color frame to a template to find known regions of text. The template counts for thresholds of background colors within a range of colors in a rectangle. Approximately 26% of the pixels in a rectangle should be within approximately 12% of the value of each channel of the background color to be identified as a region of text. This (range of colors and location) is manually provided for each TV show and reprogrammed if the TV show's graphics changes. One benefit of this algorithm is that it ignores extraneous text, such as stock tickers.

The binary image is sent to the Tesseract OCR 3.02 (block D in Fig. 1b). In our experiments, no special layout or set of words were assumed for OCR, and Tesseract OCR was run with default parameters. The output file contains any text that was recognized in the image with its (x,y) position. This information is useful because a person's name will often appear below his or her face in news stories. To prevent repetitive calls to OCR, a simple pixel comparison algorithm is also used on the output from image binarization. If the number of pixels that change is less than a threshold of 400, then the frame is not sent to OCR.

C. *Speaker Identification using Closed Captions*

In order to extract closed captions from MPEG-2 files, CCExtractor 0.65 is used (block J in Fig. 1b). Closed captions are a human-generated real-time transcript of the television show with speaker transitions marked with a “>>” prompt, story changes or multiple speakers marked with a “>>>” prompt, and timing information. Closed caption text is displayed approximately six seconds after the text is spoken. The Federal Communications Commission (FCC) requires every TV show broadcast in the United States to include closed caption information. For the TV news shows that were processed, most commercials appear with centered closed captioned text. This allows easy identification of commercials.

Digital Object Indentifier 10.1109/MMUL.2016.3 1070-986X/$26.00 2016 IEEE

The closed captions transcript is examined for hints about who is speaking using parsing 18 rules. These rules identify commonly used text which appears near a person's name that may indicate the current, previous, or next speaker. For example, the transcript may contain “I'm Dan Simon,” “Dan, thank you,” or “What do you think, Dan?” Sample parsing rules are: (1) If "I'm" starts a sentence followed by a proper name, then the proper name is speaking, (2) If a proper name starts a sentenced followed by "thank you", then the previous speaker is the proper name, and (3) If "Here's" starts a sentence followed by a proper name, then the next speaker is the proper name. Currently, when the rules conflict, the last found rule is used. These 18 closed caption rules seem to work better on news anchors and reporters rather than interviewees and subjects who appear less often. This is expected as most rules cover the interaction between news anchors and reporters. The most common name ***(ccName)*** for each speaker transition segment with the same speaker number is copied to those speaker transition segments.

D. Speaker Identification using Speaker Diarization

FFmpeg is used to extract a mono WAV audio file from the MPEG-2 file (block H in Fig. 1b). LIUM_SpkDiarization 4.2 processes the WAV file and identifies whenever the same speaker is talking throughout the audio file (block I in Fig. 1b). We used default settings of LIUM speaker diarization, which applies hierarchical GMM-based speaker clustering using the cross entropy measure to merge clusters.

Names that appear on screen during speaker diarization segments are stored as ***audioName*** for each segment. The OCR output is sent through the Stanford CoreNLP 3.2 for parts-of-speech parsing and named entity recognition. Additional parsing rules identify which text is naming the talking person and which text is a story heading. The CoreNLP misses some names so an additional list of names is used. Each diarization segment has a speaker number that identifies audio segments with the same speaker. The most common ***audioName*** for all speaker diarization segments with the same speaker number is added to those diarization segments.

E. Labeling the Speaking Person

The transcriber processes all of the above information to label the speakers (block K in Fig. 1b). Since we label speaker transitions in closed captioning, speaker transitions of the closed captioning are aligned with the speaker diarization segments. If the starting time of a speaker transition occurs during a speaker diarization segment, the speaker transition can be mapped to the speaker diarization segment. Speaker diarization provides **audioName** as a candidate name. In addition, speaker transition in closed captioning has **ccName** as a candidate.

By detecting mouth movements, our system can detect which face (or person) is talking. The name that appears on the screen while a person is talking is a candidate **faceName.** Talking instants are aligned with speaker transitions in closed captioning. Using face recognition, clustering, and talking recognition, the most common name on the screen is stored as **faceName** as a candidate name for the speaker in closed captioning.

Finally, the transcriber uses weighted majority voting to identify the speaker name at each speaker transition. For each speaker transition st_i in *ST*, a WeightedMajorityVoting function is called to choose the most common name from the *audioName*, *faceName*, and *ccName*. In case of ambiguities, the *audioName* has the highest weight. The *faceName* has the next highest weight, and *ccName* has the lowest weight. A voted name is returned for the speaker transition.

IV. Experimental Results

We have performed our experiments on a Ubuntu Linux 12.04 Intel 3GHz quad-core PC computer. Four episodes each of CNN Newsroom from analog cable were processed. Each CNN episode is 60 minutes with commercials in standard-definition TV, requiring about 95 minutes to process. Table I shows some sample speaker name output for the speaker transitions. Sometimes, a voted speaker name could not be provided. This is due to 1) there was not a name on screen when the speaker was talking, 2) a name was not found from closed captions, or 3) incorrect voice clustering due to incorrect speaker diarization. This can be seen in the first row in Table I. In the second row, the incorrect *ccName* is outvoted by the correct *audioName*, because the *audioName* has a higher weight. In the third row, the correct *ccName* was chosen because it is the only name. In the fourth row, the incorrect name is chosen because of a boundary problem, where the name is actually from the previous or next speaker transition. In the fifth row, the correct name is chosen from both the *audioName* and the *faceName*. In the sixth row, the correct name is chosen as the *audioName* and *faceName* outvote the *ccName*.

TABLE I

SAMPLE SPEAKER NAMES FOR SPEAKER TRANSITIONS FOR

CNN NEWSROOM, 2013, MAY 5, 12:00 PM

Row number	Ground-truth	Voted	audioName	talking-Name	ccName
1	WOLF BLITZER				
2	WOLF BLITZER	WOLF BLITZER	WOLF BLITZER		WELL, DAN,
3	WOLF BLITZER	WOLF BLITZER			WOLF BLITZER
4	JAMES CARANO	RICARDO ENRIOUEZ	RICARDO ENRIOUEZ	RICARDO ENRIOUEZ	
5	RICARDO ENRIOUEZ	RICARDO ENRIOUEZ	RICARDO ENRIOUEZ	RICARDO ENRIOUEZ	
6	WOLF BLITZER	WOLF BLITZER	WOLF BLITZER	WOLF BLITZER	WELL, DAN,

TABLE II

ACCURACY FOR SPEAKER IDENTIFICATION

File	true	false	miss	# of transitions	# of speakers	# of single appearances	Max appearance for any person
CNN Newsroom, 2013, April 6, 1:00 pm	68.0%	9.6%	22.4%	125	21	5	55 (Fredricka Whitfield)
CNN Newsroom, 2013, April 7, 4:00 pm	56.6%	36.8%	6.6%	136	25	9	69 (Fredricka Whitfield)
CNN Newsroom, 2013, April 24, 10:00 am	55.2%	23.8%	21.0%	105	18	5	49 (Anderson Cooper)
CNN Newsroom, 2013, May 5, 12:00 pm	74.5%	6.9%	18.6%	102	29	16	46 (Wolf Blitzer)
Average	63.6%	19.3%	17.2%				

Table II shows the accuracy of speaker identification for all CNN Newsroom shows. Average correct speaker identification percentage was 63.6%. Average wrong speaker identification percentage was 19.3%. Average unknown speaker percentage was 17.2%. When a voted speaker name could be output, average correct speaker identification percentage was 76.7%, and average wrong speaker identification percentage was 22.6%. Table III has the accuracy for speaker identification, grouped by name for CNN Newsroom, May 5, 2013, 12:00 pm. While speakers that appear frequently such as Wolf Blitzer is detected with 87% accuracy, we still achieve accuracy of 66% for a group 28 people (16 of them appearing once) even if the most frequent person is ignored.

TABLE III

ACCURACY FOR SPEAKER IDENTIFICATION, GROUPED BY NAME

		Voted (in %)			audioName (in %)			faceName (in %)			ccName (in %)		
Name	transitions	true	false	miss	true	false	miss	true	false	miss	true	false	miss
WOLF BLITZER	46	87	0	13	**78.3**	0	21.7	58.7	4.3	37	6.5	80.5	13
TED ROWLANDS	8	25	0	75	**25**	0	75	0	25	75	0	25	75
DAN SIMON	7	100	0	0	42.9	0	57.1	42.9	0	57.1	**57.1**	28.6	14.3
CHRISTINE ROMANS	5	60	20	20	0	0	100	0	0	100	**60**	20	20
FAWAZ GERGES	3	100	0	0	**100**	0	0	**100**	0	0	0	0	100
JUROR	3	0	66.7	33.3	0	0	100	0	0	100	0	66.7	33.3
ANDY SCHOLES	2	100	0	0	**100**	0	0	50	0	50	0	0	100
CMDR. MIKE MASKARICH	2	0	0	100	0	0	100	0	0	100	0	0	100
DR. JOSH ADLER	2	100	0	0	**100**	0	0	**100**	0	0	0	0	100
EMILY SCHMIDT	2	100	0	0	**100**	0	0	**100**	0	0	0	0	100
GREG MCBRIDE	2	100	0	0	**100**	0	0	**100**	0	0	0	0	100
RICARDO ENRIOUEZ	2	100	0	0	**100**	0	0	**100**	0	0	0	0	100
ZAIN ASHER	2	100	0	0	**100**	0	0	**100**	0	0	0	100	0
CHIEF JUDGE BELVIN PERRY JR.	1	100	0	0	**100**	0	0	**100**	0	0	0	0	100
CHIEF MICHAEL KEEFE	1	100	0	0	**100**	0	0	**100**	0	0	0	100	0
JAMES CARANO	1	0	100	0	0	100	0	0	100	0	0	0	100
JO-ELLAN DIMITRIUS	1	0	100	0	100	0	0	100	0	0	0	0	100
JOE JOHNS	1	100	0	0	**100**	0	0	**100**	0	0	0	100	0
JOHANA PORTILLO	1	100	0	0	**100**	0	0	**100**	0	0	0	0	100
JOHN MAPES	1	100	0	0	0	0	100	0	0	100	**100**	0	0
JONATHAN HUTCHINSON	1	0	100	0	0	0	100	0	0	100	0	100	0
JORGE RICO	1	100	0	0	**100**	0	0	**100**	0	0	0	0	100
JUDGE	1	0	100	0	0	100	0	0	0	100	0	100	0
JUSTIN BEIBER	1	0	0	100	0	0	100	0	0	100	0	0	100
LEBRON JAMES	1	0	0	100	0	0	100	0	0	100	0	0	100

Digital Object Indentifier 10.1109/MMUL.2016.3 1070-986X/$26.00 2016 IEEE

PAUL CALLAN	1	0	0	100	0	0	100	0	0	100	0	0	100
PETER STEFAN	1	100	0	0	**100**	0	0	**100**	0	0	0	0	100
RYAN MACK	1	100	0	0	**100**	0	0	**100**	0	0	0	0	100
SEN. CHARLES SCHUMER	1	100	0	0	**100**	0	0	**100**	0	0	0	0	100

The accuracy of audioName, faceName, and ccName are 59.8%, 50%, and 10.8% for Table III. Our system achieves 74.5% using our weighted voting scheme. In 11.8% of the cases audioName provided the only correct recognition while faceName provided the only correct result in 1% of the cases. Despite the low accuracy of ccName, it was the only correct method in 10.8% of the cases. In 27.4% of the cases no method were able to identify the person correctly. There are a few reasons for low accuracy of ccName: 1) during conversation, the speaker may start using only the first or last name of the other person rather than the full name, 2) the title of a person may be omitted when searching proper names in the closed captions, 3) misspelling of a name (e.g., 'CHRISTINE ROMANS' was detected as 'CHRISTINE ROMAN'), and 4) referral to a third person during the speech. Errors from ccName were considered as false detection even if they were partially correct. AudioName and faceName provided similar performances and they both benefit from the OCR. The OCR usually worked well but there were a few times it made minor mistakes. For example, 'ANDY SCHOLES' was detected as 'ANDV SCHOLES' and 'JO-ELLAN DIMITRIUS' was detected as 'JO~ELLAN DIMITRIUS.' Both cases were assumed to be correct detection in our analysis. The lower performance for talking name results from absence of text when a person speaks. There was only one other case where faceName was false but audioName was correct.

Our proposed method works if the TV network displays names consistently and reporters introduce themselves. We believe displaying names and introducing people are necessary for our unsupervised speaker recognition. There were a few adjustments made to algorithms to suit them for CNN news video. OCR software looks for a specific region and background for text to extract names on the scene. Our style for the closed captioning is similar to the format specified at "Realtime Broadcast Captioning: Recommended style and Format Guidelines for US Programming" by NCRA (ncra.org) Captioning Community of Interest. If there are significant deviations, they should be incorporated for closed captioning parsing.

Digital Object Indentifier 10.1109/MMUL.2016.3 1070-986X/$26.00 2016 IEEE

V. Conclusion and Future Work

This paper presents an unsupervised method to identify speaker names for closed captions in TV news video files without using external input. Our method needs to find and recognize faces, determine if they speak, check whether names appear on screen, parse closed captions for possible names, and cluster speakers based on their voice similarity. For this challenging problem, we were able to achieve 63.6% accuracy for CNN news shows without training our system for any speaker.. A U.S. Patent was filed in March 2015.

The accuracy can be enhanced by improving each component of our algorithm. The biggest improvement in accuracy may be from maintaining a database of face and audio speaker data that were automatically recognized and labeled by our system. This allows past information to be incorporated into future learning decisions, reducing the required information in the future video files and increasing accuracy. Moreover, improving the parsing rules should increase the accuracy of the closed caption and OCR text processing. Instead of manually creating parsing rules, statistical algorithms could learn parsing rules from a pre-labeled dataset. An optimum order of rules can be analyzed to determine speakers when rules conflict.

References

1. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Comput. Vision and Pattern Recognition (CVPR), Proc. of the 2001 IEEE Conf. on*, 2001, pp. 511-518.
2. Luxand, "FaceSDK," 2014, Available: http://www.luxand.com.
3. M. Turk and A. Pentland, "Eigenfaces for recognition," *J. of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.
4. P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *Pattern Anal. and Mach. Intell., IEEE Trans. on*, vol. 19, no. 7, pp. 711-720, 1997.
5. T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51-59, 1996.
6. T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *Comput. Vision - ECCV 2004*: Springer, 2004, pp. 469-481.
7. T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Comput. Vision and Image Understanding*, vol. 61, no. 1, pp. 38-59, 1995.
8. M. Cox, J. Nuevo, J.Saragih and S. Lucey, "CSIRO Face Analysis SDK", *AFGR*, 2013.

9. M. R. Lyu, J. Song, and M. Cai, "A comprehensive method for multilingual video text detection, localization, and extraction," *Circuits and Syst. for Video Technology, IEEE Trans. on*, vol. 15, no. 2, pp. 243-255, 2005.

10. "Tesseract-ocr," 2015, Available:https://github.com/tesseract-ocr.

11. Stanford, "CoreNLP," 2014, Available: http://nlp.stanford.edu/software/corenlp.shtml.

12. S. Meignier and T. Merlin, "LIUM SpkDiarization: an open source toolkit for diarization," *CMU SPUD Workshop*, 2010.

13. M. Everingham, J. Sivic, and A. Zisserman, "Taking the bite out of automated naming of characters in TV video," *Image and Vision Computing*, vol. 27, no. 5, pp. 545-559, 2009.

14. J. Sang and C. Xu, "Robust face-name graph matching for movie character identification," *Multimedia, IEEE Trans. on*, vol. 14, no. 3, pp. 586-596, 2012.

15. Y.-F. Zhang, C. Xu, H. Lu, and Y.-M. Huang, "Character identification in feature-length films using global face-name matching," *Multimedia, IEEE Trans. on*, vol. 11, no. 7, pp. 1276-1288, 2009.

16. Stein, Daniel, Evlampios Apostolidis, Vasileios Mezaris, Nicolas de Abreu Pereira, Jennifer Müller, Mathilde Sahuguet, Benoît Huet, and Ivo Lasek. “Enrichment of News Show Videos with Multimodal Semi-Automatic Analysis,” 2012.

17. Stein, D., A. Öktem, E. Apostolidis, V. Mezaris, José Luis Redondo García, Raphaël Troncy, Mathilde Sahuguet, and Benoît Huet. “From Raw Data to Semantically Enriched Hyperlinking: Recent Advances in the LinkedTV Analysis Workflow,” 2013.

18. H. Jiang, A. S. Helal, A. K. Elmagarmid, and A. Joshi, "Scene change detection techniques for video database systems," *Multimedia Syst.*, vol. 6, no. 3, pp. 186-195, 1998.

Daniel N. Woo received the B.S. degree in 1993 and M.S. degree in 2014 in computer science from the University of Alabama in Huntsville.

Ramazan S. Aygün is currently an Associate Professor in Computer Science Department, University of Alabama in Huntsville. His research interests include multimedia databases and video processing.