Automated Estimation Of Time Codes For Captioning Digital Media

Daniel P. Harvey II

Eastern Illinois University

This research is a product of the graduate program in Technology at Eastern Illinois University. Find out more about the program.

Recommended Citation
http://thekeep.eiu.edu/theses/692

This Thesis is brought to you for free and open access by the Student Theses & Publications at The Keep. It has been accepted for inclusion in Masters Theses by an authorized administrator of The Keep. For more information, please contact tabruns@eiu.edu.
AUTOMATED ESTIMATION OF TIME CODES FOR CAPTIONING DIGITAL MEDIA

HARVEY
THESIS REPRODUCTION CERTIFICATE

TO: Graduate Degree Candidates (who have written formal theses)

SUBJECT: Permission to Reproduce Theses

The University Library is receiving a number of request from other institutions asking permission to reproduce dissertations for inclusion in their library holdings. Although no copyright laws are involved, we feel that professional courtesy demands that permission be obtained from the author before we allow these to be copied.

PLEASE SIGN ONE OF THE FOLLOWING STATEMENTS:

Booth Library of Eastern Illinois University has my permission to lend my thesis to a reputable college or university for the purpose of copying it for inclusion in that institution's library or research holdings.

Daniel P. Harvey II  4/23/2008

Author's Signature  Date

I respectfully request Booth Library of Eastern Illinois University NOT allow my thesis to be reproduced because:

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________


________________________________________________________________________

Author's Signature  Date

This form must be submitted in duplicate.
Automated Estimation of Time Codes
for Captioning Digital Media

BY
Daniel P. Harvey II

THESIS
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Master of Science in Technology

IN THE GRADUATE SCHOOL, EASTERN ILLINOIS UNIVERSITY
CHARLESTON, ILLINOIS

2008

I HEREBY RECOMMEND THAT THIS THESIS BE ACCEPTED AS FULFILLING
THIS PART OF THE GRADUATE DEGREE CITED ABOVE

4/24/08
DATE

Ping Lin
THESIS DIRECTOR

May 5, 2008
DATE

DEPARTMENT/DEAN'S SCHOOL HEAD
Estimation of Time Codes for Captioning

Thesis Committee

Peter Ping Liu
Ph.D., OCP, P.E., C.Q.E. and C.S.I.T.
Professor
Thesis Director
Coordinator of Graduate Studies
School of Technology

Samuel A. Guccione
Ed.D., C.S.I.T.
Associate Professor
School of Technology

Rigoberto Chinchilla
Ph.D.
Assistant Professor
School of Technology

4/24/08
Date

4/24/08
Date

4/26/08
Date
Abstract

Captioning provides accessibility to media resources for deaf and hearing-impaired persons and is mandated by federal and state laws such as the Americans with Disabilities Act. However, the process of creating synchronized captions for media is time consuming and labor intensive. Consequently, many content providers still do not incorporate captions into media presentations on the web.

In this research, algorithms were developed to automate part of the captioning process by estimating the timing of captions for web-based audio and video files using plain-text transcripts and their corresponding audio recordings. Recordings used in this study were of professional speakers/readers from American English radio and television broadcasts and of non-professional speakers reading text from a novel. The text transcripts were divided into sentences. The duration of each sentence was initially estimated from the number of characters in each sentence as a proportion of the total recording time. The locations and durations of pauses in the audio recordings were compiled by scanning for regions of low amplitude. It was found that the RMS amplitude of each audio file performs adequately as a threshold between silence and speech for captioning.

Statistically, pause durations at the ends of sentences are significantly greater than those within sentences. This observation was used to match the ends of sentences in the text to pauses in the audio track. In order to successfully distinguish between within-sentence pauses and end-of-sentence pauses on the basis of duration, it is advantageous to utilize data over localized portions of a file rather than over the entire audio file. When the results of the automated matching were ambiguous, a manual feedback mechanism
was utilized to further improve the accuracy of the algorithms. For the media files tested in this study, the algorithm accurately estimated the timing of 96% of the captions within 0.5 seconds.
Acknowledgements

I would like to express my sincere thanks to Dr. Peter Ping Liu for his guidance throughout the research and writing process. His abilities to ask the right questions and suggest new directions of thought have added much to this work.

Thank you to Dr. Sam Guccione and Dr. Rigoberto Chinchilla for serving on my Thesis Committee. Their suggestions and feedback are greatly appreciated.

Thank you to Dr. Michael Hoadley, Director of the Center for Academic Technology Support, for his support of this research and of my graduate program.
Table of Contents

Chapter 1. Introduction ..................................................................................................... 1

1.1 Need for Automated Captioning ......................................................................... 1

1.2 Statement of the Problem .................................................................................. 2

1.3 Statement of the Purpose .................................................................................. 2

1.4 Research Hypothesis ......................................................................................... 3

1.5 Research Questions .......................................................................................... 3

1.6 Assumptions ......................................................................................................... 4

1.7 Limitations ........................................................................................................... 4

1.8 Delimitations ....................................................................................................... 5

1.9 Definition of Terms ........................................................................................... 5

Chapter 2. Literature Review ....................................................................................... 7

2.1 Digital Media ....................................................................................................... 7

2.2 Captioning Techniques and Practices ............................................................... 7

2.3 Variants of XML Used for Captioning ............................................................... 9

2.4 Current Tools ..................................................................................................... 22

2.5 Structure of Speech ........................................................................................... 23

2.6 Pauses ................................................................................................................ 24

2.7 Punctuation ......................................................................................................... 26

2.8 Silence Detection ................................................................................................ 28

2.9 Summary .............................................................................................................. 29
Chapter 3. Methods ........................................................................................................ 31

3.0 Overview ........................................................................................................ 31

3.1 Audio and Video Data .................................................................................... 31

3.2 Parsing a Text Transcript into Sentences .................................................... 32

3.3 Audio Data Analysis ...................................................................................... 33

3.4 Web Interface .................................................................................................. 36

3.5 Timecode Estimation Algorithms ................................................................... 39

3.5.1 Global Threshold Method ........................................................................... 40

3.5.1.1 Character Count Weighting (CCW) Algorithm ................................. 40

3.5.1.2 Character Count Weighting and Punctuation Pauses
(CCW + PP) Algorithm ................................................................................ 41

3.5.1.3 Character Count Weighting, Punctuation Pauses,
and Pauses Between Words (CCW + PP + PBW) Algorithm ..... 45

3.5.1.4 Feedback Using Temporal Proximity ................................................ 47

3.5.2 Local Maxima Method .............................................................................. 49

3.5.2.1 Automatic Correction Algorithm ..................................................... 51

3.5.2.2 Manual Reset Algorithm ..................................................................... 52

3.6 Effect of Amplitude Threshold on Timecode Estimation Error ..................... 53

3.7 Data Analysis .................................................................................................. 54

3.8 Development of Code to Generate Captioned Media ................................... 54

3.9 Summary .......................................................................................................... 55
List of Figures

Figure 1. Sample SMIL Code for RealPlayer

Figure 2. Sample RealText File

Figure 3. Preferences Window in RealPlayer

Figure 4. Sample SMIL Code for QuickTime

Figure 5. Sample QuickTime Text File

Figure 6. Sample .asx file for Windows Media Player

Figure 7. Sample SAMI file for Windows Media Player

Figure 8. Sample XML captions file for Flash Player

Figure 9. Action Script to insert into Flash document to synchronize captions and Flash Video.

Figure 10. Screen shot of vol1ch1.wav used to measure recording noise.

Figure 11. Screen shot of vol1ch1.wav as seen in Audacity.

Figure 12. Web interface for the manual determination of caption timecodes.

Figure 13. Sample output of transcript parsing and text analyses for car.avi.txt.

Figure 14. Sample output of caption times estimated from the text analyses and the Temporal Proximity Feedback algorithm from the Global Threshold Method.

Figure 15. Sample output of transcript parsing and local maxima analysis.

Figure 16. Pause Duration as a function of Time for media file car.wav.

Figure 17. Pause Duration as a function of Time for media file basraLibrary.wav.

Figure 18. Effect of threshold factor on average error magnitude for vol1ch1.wav
Figure 19. Average Error of Global Threshold Text-based Estimation as a function of Standard Deviation of End-of-Sentence Pause Duration and the Total Pause Duration Error due to Incorrect Pause Selection.

Figure 20. Effect of minimum search range on timecode error magnitude in vollch1.wav.

Figure 21. Timecode estimation error as a function of percentage of estimated duration of previous caption (minimum of 1.5 s) in vollch1.wav.

Figure 22. Timecode estimation error for basraLibrary.wav.

Figure 23. Maximum Amplitude histogram for elections.wav.

Figure 24. Effect of amplitude threshold on timecode estimation error, car.wav.
List of Tables

Table 1. *Reported Values of Pause Duration for Various Syntactic Boundaries in Read Speech*

Table 2. *Media Files Used in the Present Study.*

Table 3. *Fields in database tables used in web interface.*

Table 4. *Differences in Durations of Pauses Corresponding to Within-Sentence Punctuation and End-of-Sentence Punctuation.*

Table 5. *Mann-Whitney U Test on Pause Duration Normalized by Total Number of Pauses Associated with Punctuation Points.*

Table 6. *Differences in Durations of Pauses Corresponding to Within-Sentence Punctuation and End-of-Sentence Punctuation for Segments of Non-ideal Audio Files.*

Table 7. *Mann-Whitney U Test on Pause Duration Normalized by Total Number of Pauses Associated with Punctuation Points for Segments of Non-ideal Audio Files.*

Table 8. *Average error magnitude for text analysis algorithms and global pause matching.*

Table 9. *Average Magnitude of Error for Local Maxima Method.*

Table 10. *Average Error and Improvement Provided by Manual Feedback, Manual Input Requests, Number of Captions, Manual Feedback Rate.*

Table 11. *Peak Amplitude from Histograms and Manually Measured Recording Noise from Audio Files (full scale = 32767).*

Table 12. *Background Noise and Global RMS from Audio Files (full scale = 32767).*
Table 13. *Average Timecode Estimation Error in Local Maxima Method with Manual Feedback Using Upper Onset of Histogram Peaks and Global RMS as Amplitude Threshold*
Contents of the Supplemental CD

Documents

- Copy of Thesis in MS Word

Examples of Captioned Media

- Flash
- QuickTime
- Windows Media Player
- RealPlayer

Instructions: Insert CD into CD drive. On a PC, the file menu.html will automatically open in a web browser window. On a Mac, double-click on the CD-Rom icon and open the file menu.html with a browser. You can then use the links on the web page to view examples of captioned media created using the process developed in this research.
Chapter 1

Introduction

1.1 Need for Automated Captioning

The presentation of information on the World Wide Web relies increasingly on multimedia technologies. The audio component of these media presentations on the web remains largely inaccessible to persons who are Deaf or Hard of Hearing (Canadian Network for Inclusive Cultural Exchange, 2004; Solomon, 2001). Adults reporting some level of hearing difficulty comprise 17% of the population of the United States or approximately 35.6 million people (Lucas, Schiller, & Benson, 2004). Regulations for Federal and State Governmental websites require captions for web-based multimedia (General Services Administration, 2002; Illinois Technology Office, 2002).

Captioning is the process of converting the narration, dialogue, music and sound effects of a media production into text and displaying it (National Captioning Institute, n.d.). Captions provide access to web audio and video for persons with hearing disabilities. Although captioning is primarily intended for those who cannot receive the benefit of audio, it has been found to greatly help individuals who can hear audio, but whose primary language is different from that of a media presentation (WebAIM, n.d.-a). There are approximately 46.9 million persons in the United States for whom English is a second language that self-report that they speak English less than “very well” (U.S. Census Bureau, 2003). Even though there is probably some overlap between these two groups, there are tens of millions of people in the United States alone that benefit from captioned media.
Captions for web-based media are implemented by creating text captions that are separate from, yet synchronized with the media using a media markup language. Users can then set a preference in a media player to display text captions if they are available. Existing tools facilitate the captioning process by generating the code that integrates and synchronizes a media file with text captions. However, captioning is time and labor intensive because it requires manually generating a text transcript and parsing it into captions as well as manually determining time codes. If the process can be automated even further by automating the parsing of a text transcript into captions and the determination of time codes, web developers will be more likely to incorporate captions in web-based media presentations.

1.2 Statement of the Problem

Captions for web-based multimedia are required by Federal and State Governmental regulations. The production of captions that meet the criteria of synchronization, equivalence, and accessibility is very time consuming. Most existing services and products for captioning require manual determination of timing for captions and are, therefore, labor-intensive and expensive. There is a need to automate the process of converting a full-text transcript into synchronized captions.

1.3 Statement of the Purpose

The purpose of this study is to develop algorithms to automate the determination of time codes for captions from a full-text transcript. The general approach will involve estimating the temporal structure of speech according to data collected from text transcripts and digital audio files. The timing of speech in an audio clip will be initially estimated by assuming that the speech is uniformly distributed throughout the media clip
and that each letter in the transcript takes the same amount of time to articulate. The timing estimate will then be refined by accounting for pauses in speech that correspond to punctuation marks in the transcript. The punctuation marks in the text will be correlated to pauses in speech measured from audio data using both the duration and the relative location of the pauses in the media clip. The goal of this research is to develop ways to generate captions more effectively and efficiently, which will promote more widespread captioning of web-based media.

1.4 Research Hypothesis

Text transcripts and digital media files provide information that will guide the determination of time codes for captioning. There is a correlation between measurable pauses in speech and punctuation marks in a transcript and the correlation can be used to satisfactorily estimate time codes for synchronized captions.

1.5 Research Questions

In order to accurately correlate measurable pauses in speech and punctuation marks in a transcript, a number of issues regarding the collection, analysis, interpretation, and combination of data from text transcripts and audio files will be investigated.

1. How can text transcripts be analyzed to provide timing information? First order approximation can be derived from the order of words in the text. The number of characters in a word or in a caption phrase can provide an initial estimate of relative timing. The possibility of further refining estimates of speech timing by using punctuation marks to locate syntactic pauses in speech and using spaces between words to locate inter-lexical pauses will be investigated.
2. How can actual locations and durations of pauses be measured from a digital media file? Sound data are stored in a binary file formats. Generally, there is a segment of the file that contains formatting information followed by the actual sound data. The data is in the form of amplitude vs. time. The feasibility of accurately measuring pauses in speech by searching for extended regions of low amplitude in digital audio recordings will be investigated.

3. How can audio and transcript data be combined to improve accuracy of time code estimation? The correlation of measured pauses from digital media files to duration and location estimates derived from text transcript analysis will be investigated. Specific examples include correlating the longest pauses in speech to punctuation marks that denote the ends of phrases, clauses, sentences, and paragraphs in a transcript and correlating the timing of speech between pauses to the length of its corresponding text transcript.

1.6 Assumptions

1. Manually determined time codes will be accurate enough to serve as a benchmark for estimated time codes.

2. The time it takes to articulate a phrase is proportional to the number of letters in the transcript of that phrase.

3. Major pauses in speech correspond to punctuation marks in a transcript.

1.7 Limitations

1. Grammatically correct transcripts were manually generated.

2. Background noise level of recordings cannot be completely eliminated or controlled.
1.8 Delimitations

1. Recordings were of professional speakers/readers from American English radio and television broadcasts and of non-professional speakers reading text from a novel. This study did not address informal or conversational speech.

2. Digital recordings with a high signal-to-noise ratio and with minimal background sounds were used in this study.

3. Captions were produced in post-production; therefore, they will not be applicable to live streaming media.

4. Media file formats utilized in this study were .wav audio and. avi audio/video.

1.9 Definition of Terms

Caption duration: The caption duration is the amount of time each caption appears and is the difference between the caption start time and the next caption start time, or in the case of the last caption, the difference between the caption start time and the end of the media clip.

Caption start time: The caption start time is when each caption appears and is synonymous with the time code. The caption start times correspond to pause end times.

Captioning: Captioning is the process of converting the narration and dialogue of an audio or video production into text and displaying it (National Captioning Institute, n.d.).

Clip Time: The clip time is the duration of the entire media file.

Inter-lexical pauses: Inter-lexical pauses are pauses between words; typically >200 ms, but always significantly longer than syllables and intra-segmental pauses; these provide larger scale structure to speech (Zellner, 1994).

Intra-segmental pauses: Intra-segmental pauses are pauses within words, typically <200 ms (Zellner, 1994).
Pause duration: The pause duration is the amount of time each region of low amplitude lasts and is the difference between the pause start time and pause end time.

Pause end time: Pause end times are when transitions from low amplitude to high amplitude regions in the audio file occur.

Pause start time: Pause start times are when transitions from high amplitude to low amplitude regions in the audio file occur.

Punctuation marks: Punctuation marks are written symbols that do not correspond to either phonemes (sounds) of a spoken language, nor to lexemes (words and phrases) of a written language, but which serve to organize or clarify written language and is used, in printing and writing, to imitate speech (Complete Translation Services, Inc., 2004).

Rhythm: Rhythm is how words and syllables are grouped in time (Netsell, 1973).

Segments: Segments are consonants and vowels (Campbell, 1992).

Speech prosody: Speech prosody is extra-lingual information in speech such as timing and intonation patterns that convey structural, semantic, and functional information (Shriberg, Stolke, Hakkani-Tur & Tur, 2000).

Speech rate or Tempo: Speech rate or tempo is the number of words spoken in a given amount of time (Kent & Read, 1992; Trouvain, 2002).

Speech ratio: The speech ratio is the ratio of actual speech time to total speaking time (Furui, 2001).

Suprasegmentals: Suprasegmentals are features of speech that extend beyond an individual segment (Borden, Harris, & Raphael, 2003).

Timecode: Timecodes are information paired with each caption that determines when it appears (National Center for Accessible Media, 2000).

Total speaking time: Total speaking time is the sum of actual speech or articulation time and pause time (Walker, 1988).
2.1 Digital Media

A common file structure for digital audio is the WAVE format. The WAVE format uses pulse code modulation (PCM) (Dillon & Leonard, 1998; Gibbs & Tsichritzis, 1995). PCM samples a waveform at a constant rate of thousands of times a second to convert analog audio to digital audio. PCM does not involve any data compression (Furui, 2001) and is the best established and the most implemented digital coding system. PCM is versatile and is widely accepted as a standard against which to calibrate other approaches to waveform digitization (Jayant & Noll, 1984). The wave file format contains 44 bytes of format data including the file size, file type, channels (mono or stereo), sample rate, and bits per sample (8 or 16). Eight-bit samples are stored as unsigned bytes, ranging from 0 to 255. Sixteen-bit samples are stored as signed integers, ranging from -32768 to 32767. The actual audio data starts at 44 bytes into the file (Weber, n.d.). By measuring the wave amplitude in a digital audio file, pause locations and durations can be measured by searching for extended regions of low amplitude (Horii, 1983). The corresponding time of any data point may be determined from counting the data points because the sampling rate of the waveform is known.

2.2 Captioning Techniques and Practices

There are existing governmental guidelines and industry standards for the accessibility of web based media. Regulations for Federal Government websites state that, "equivalent alternatives for any multimedia presentation shall be synchronized with the presentation" (General Services Administration, 2002). The World Wide Web
Estimation of Time Codes for Captioning

The Consortium (W3C) suggests that for time-based multimedia presentations (e.g., a movie or animation), equivalent alternatives (e.g., captions or auditory descriptions of the visual track) should be synchronized with the presentation (Chisholm, Vanderheiden & Jacobs, 1999). The Illinois Web Accessibility Standards require Illinois governmental agencies to provide synchronized captions for multimedia containing speech (Illinois Technology Office, 2002). According to these guidelines, captions should appear at approximately the same time that the corresponding audio is available and the content of the captions should be equivalent to that of the spoken word. Also, the captions should be readily accessible and available to those who need it using widely available technology such as browser plug-ins and media players (WebAIM, n.d.-a).

Generally, commas, periods, and dashes are the only marks used in the transcription of spoken discourse (Nunberg, 1990). Written English relies on word order to make meaning clear. When word order alone is not sufficient, punctuation is used. Because, acceptable and understandable speech may consist of broken sentences, incomplete sentences, run-on sentences, and other constructions not acceptable when originated as written language; transcription of speech sometimes requires punctuation that is unique to the captioning process. Double hyphens are used to offset an aside or nonessential information. Capital letters are used to indicate screaming. When captioning a spelled word, hyphens are used to separate the letters. Ellipses marks are used when there is a significant pause within a caption (Captioned Media Program, 2004).

According to governmental guidelines and industry standards, captions should appear at approximately the same time that the corresponding audio is available and the
content of the captions should be equivalent to that of the spoken word (Chisholm, Vanderheiden & Jacobs, 1999; General Services Administration, 2002; Illinois Technology Office, 2002). Also, captions should be readily accessible and available to those who need it using widely available technology such as browser plug-ins and media players (WebAIM, n.d.-a). The National Association for the Deaf suggests a caption presentation rate of 150-160 words per minute (wpm) (Captioned Media Program, 2004). This corresponds to observations of reactions to caption speed that showed that 145 wpm is comfortable for viewers but that speeds exceeding 170 wpm presents difficulty for viewers. As hearing ability increased, preference for speed decreased, which was attributed to less practice reading captions (Jensema, 1998). A study of the statistical characteristics of closed captions on 205 television programs showed that mean caption speed was 141 words per minute with a maximum of 231 wpm and a minimum of 74 wpm (Jensema, McCann, & Ramsey, 1996). A suggested maximum difference between captions and audio is 0.5 s because this is barely perceptible to a viewer (Captioned Media Program, 2004). Studies have also shown that viewers preferred display of complete sentences over sentence fragments, Helvetica rather than standard decoder font, white over yellow text, mixed case over all uppercase, and pop-on presentation rather than fade-in (Kirkland, 1999).

2.3 Variants of XML Used for Captioning

Synchronized Multimedia Integration Language (SMIL) is a text-based markup language that is used to synchronize and integrate multimedia (Kennedy & Slowinski, 2002; Hoschka, 1998) and can therefore be used as a framework to synchronize media clips and captions in RealPlayer. With SMIL, one can control the relative positioning,
display, and relative timing of individual media elements. A SMIL file contains references to individual media files such as streaming video, streaming audio, image files, text files, animation, and interactive elements.

An example of SMIL code for RealPlayer is shown in Figure 1. The basic structure module elements of SMIL are <smil>, <head>, and <body>. These elements serve as containers for all of the additional coding. The <smil> tag is the outer container or the entire code; the <head> and <body> elements sit within this container. The <head> element provides information that is not related to the timing of the media elements such as metainformation to describe the presentation and the layout of the presentation which determines the relative positioning of the media elements. The <body> element contains pathways to media source files and all of the information related to the timing of the presentation such as start and stop times and sequencing of media elements. For the purposes of captioning, tags referring to the media file and to the caption file are needed. The media file can be any audio or video format that will play in RealPlayer. The caption text and time codes are contained in a RealText file, as shown in Figure 2. The basic format of the code used to control caption display are a <time> tag with a begin parameter, a clear tag so that only one caption is displayed at a time, and the caption text.

Both the <head> and <body> elements can use the <switch> tag, which allows for the authoring of multimedia presentations that are tailored to individuals based on language, data rate, or other parameters. The parameter of interest for captioning is system-captions, which makes the display of captions optional based on the preferences of each viewer. To implement the system captions in RealOne Player a user would
<smil>
<head>
<meta name="title" content="SMIL Wrapper"/>
<layout>
<root-layout background-color="black" height="300" width="355"/>
<region id="videoregion" background-color="black" top="5"
left="5" height="240" width="350"/>
<region id="textregion" background-color="#000000" top="240"
left="5" height="60" width="350"/>
</layout>
</head>
<body>
<par>
<!-- VIDEO -->
<video src="car.avi" region="videoregion"/>
<!-- CAPTIONS -->
<switch>
<textstream src="car.avi.txt.rt" region="textregion" system-
language="en" system-captions="on" title="english captions"
alt="english captions"/>
</switch>
</par>
</body>
</smil>

Figure 1. Sample SMIL Code for RealPlayer
Someone watching a car accelerate toward light speed would see something very strange.

It would seem as though the car itself were getting shorter and that time for the person in the car was slowing down.

However, you wouldn't see these effects until the car began to approach the speed of light.

At ninety percent of the speed of light, the car would appear to shrink to forty four percent of its usual length.

This thought experiment answered Einstein's old question about what he would see if he traveled along with a beam of light.

He simply couldn't make the trip, for at the speed of light, length would contract to zero and time would stop.

Figure 2. Sample RealText File
Figure 3. Preferences Window in RealPlayer
choose Tools from the main tool bar and Preferences from the Tools sub-menu. In the Preferences window (Figure 3), a user would then choose Content in the category menu, go to the Accessibility portion of the window and checking the check box labeled "Use supplemental text captioning when available."

The structure of a SMIL file for QuickTime is similar to that of SMIL for RealPlayer, except for an added parameter in the SMIL tag at the beginning of the file (see Figure 4). The syntax of a QuickTime text file containing captions and their timecodes is very different from the RealText format and conforms the least to XML standards (Figure 5). The parameters for determining the visual properties of the captions are enclosed in braces and the timecodes are enclosed in square brackets preceding each caption. In RealText, fractions of seconds are expressed in decimal format while in QuickTime text; fractions of seconds are expressed as frames where 1 second equals 30 frames.

Captioning for Windows Media Player is invoked by creating a SAMI file that contains the caption texts and timecodes and an asx file which associates a media file with the SAMI file as shown in Figure 6. The SAMI file (Figure 7) conforms to XML standards and the timecodes are expressed in milliseconds. The XML captions file for Flash Player (Figure 8) is relatively simple. However, combining the captions with a Flash video requires inserting an action script (Figure 9) into a Flash document and exporting the Flash document as a Shockwave file for viewing. The process for adding captions to a Flash video is available online (South Carolina Macromedia User Group, 2007).
<smil


" qt:autoplay="true" qt:time-slider="true">

<head>
<meta name="title" content="SMIL Wrapper"/>
</head>

<layout>
<root-layout background-color="black" height="306" width="350"/>
<region id="videoregion" background-color="black" top="0"
left="0" height="240" width="350"/>
<region id="textregion" background-color="#000000" top="240"
left="0" height="66" width="350"/>
</layout>
</head>

<body>
<par>
<!-- VIDEO -->
<video src="car.avi" region="videoregion"/>
<!-- CAPTIONS -->
<textstream src="car.avi.txt.qt.txt" region="textregion" system-
language="en" system-captions="on" title="english captions"
alt="english captions"/>
</par>
</body>
</smil>

Figure 4. Sample SMIL Code for QuickTime
[00:00:02.1] Someone watching a car accelerate toward light speed would see something very strange.

[00:00:07.18] It would seem as though the car itself were getting shorter and that time for the person in the car was slowing down.

[00:00:15] However, you wouldn't see these effects until the car began to approach the speed of light.

[00:00:21] At ninety percent of the speed of light, the car would appear to shrink to forty four percent of its usual length.

[00:00:29.1] This thought experiment answered Einstein's old question about what he would see if he traveled along with a beam of light.

[00:00:36.29] He simply couldn't make the trip, for at the speed of light, length would contract to zero and time would stop.

[00:00:44.4044]

*Figure 5. Sample QuickTime Text File*
<asx version="3.0">
<abstract>This is the shows abstract</abstract>
<title>car 5000</title>
<author></author>
<copyright>(c) 2002 - company name</copyright>
<entry>
<ref href="car.avi?SAMI=car.avi.txt.smi">
<abstract>This is the clips abstract</abstract>
<title>car 5000</title>
<author></author>
<copyright>(c) 2000 - company name</copyright>
</entry>
</asx>

Figure 6. Sample .asx file for Windows Media Player
Figure 7. Sample SAMI file for Windows Media Player
<captions>

<caption start="2.32651">Someone watching a car accelerate toward light speed would see something very strange.</caption>

<caption start="7.61005">It would seem as though the car itself were getting shorter and that time for the person in the car was slowing down.</caption>

<caption start="15.0159">However, you wouldn't see these effects until the car began to approach the speed of light.</caption>

<caption start="21.0138">At ninety percent of the speed of light, the car would appear to shrink to forty four percent of its usual length.</caption>

<caption start="29.3327">This thought experiment answered Einstein's old question about what he would see if he traveled along with a beam of light.</caption>

<caption start="36.9761">He simply couldn't make the trip, for at the speed of light, length would contract to zero and time would stop.</caption>

</captions>

Figure 8. Sample XML captions file for Flash Player
var aCaptions:Array;

var xmlCaptions:XML = new XML();
xmlCaptions.ignoreWhite = true;

xmlCaptions.onLoad = function():Void {
aCaptions = this.firstChild.childNodes;
}

for(var i:Number = 0; i < aCaptions.length; i++) {
cmpInstance.addCuePoint(i, aCaptions[i].attributes.start);
}

// Load the XML data.
xmlCaptions.load("despotism.xml");

// Tell the MediaPlayback instance what to play.
cmpInstance.contentPath = "flashVideo.flv";

cmpInstance.controllerPolicy = "on";

cmpInstance.addEventListener("cuePoint", onCuePoint);

tCaptions = this.createTextField("tCaptions", 1, cmpInstance.x, cmpInstance.y + cmpInstance.height, cmpInstance.width, 50);
tCaptions.multiline = true;
tCaptions.wordWrap = true;

function onCuePoint(e:Event):Void {

tCaptions.text = aCaptions[e.cuePointName].firstChild.nodeValue;
}

Figure 9. Action Script to insert into Flash document to synchronize captions and Flash Video.
The W3C suggests QuickTime 3.0, SMIL (Synchronized Multimedia Integration Language) with RealPlayer, or SAMI (Microsoft's Synchronized Accessible Media Interchange) with Windows Media Player as formats to add captions to media (Chisholm, Vanderheiden & Jacobs, 2000). The State of Illinois specifies that captions should be implemented using SMIL to synchronize the display of text from a transcript with the video, although as a less desirable alternative, captions can be added to a standard video recording and then converted to a web form (Illinois Web Accessibility Standards, 2002). In terms of accessibility, RealPlayer and Windows Media Player have been rated higher than QuickTime (WebAIM, n.d.-b). However, SAMI is supported solely by Microsoft products (Microsoft, 2003). In fact, Microsoft's involvement with the W3C in the development of SMIL 2.0 specifications (Ayars, et. al., 2001) suggests that SAMI may eventually be phased out. RealPlayer with SMIL has the advantages of its greater relative accessibility (WebAIM, n.d.-b), its compliance with State of Illinois specifications (Illinois Web Accessibility Standards, 2002), and its compatibility with a wider range of products and operating systems (Microsoft, 2003). RealPlayer is available via free download from the web at http://www.real.com; however, in order to play RealPlayer with SMIL without downloading the media, text and SMIL files, it is necessary to stream the files on a Real Media Server, which requires a yearly licensing fee. Windows Media Server is included for free with the Windows Web Server and can be set up to play within the Internet Explorer browser and with the appropriate plug-ins, can play embedded in web pages for Firefox and Netscape (Microsoft, 2007; Microsoft, 2008). QuickTime and Flash Player have the advantage of being usable across multiple browsers and across operating systems (PC vs. Mac). Recent data shows that Flash Player is present on 98%
of all computers and QuickTime is present on 67% of all computers (Adobe Systems Incorporated, 2007). Because each of these media players has advantages and disadvantages, captioning tools developed in this study will support all four of them.

2.4 Current Tools

Software is available to assist in the creation of captioned web-based media. MAGpie is an application provided by the National Center for Accessible Media and may be downloaded from the web at no cost (National Center for Accessible Media, n.d.-a). The Computer Prompting and Captioning Company (CPC) sells a series of hardware/software packages for real time and post-production captioning (Computer Prompting and Captioning, n.d.). MAGpie and the CPC systems allow caption text to be imported from a properly formatted text transcript; however, time codes need to be determined manually. Consequently, it takes thirty to forty minutes to caption every ten minutes of audio. There are a number of commercially available software products that are available to edit SMIL code (Michel, 2004) and to specifically assist in the creation of captions for web-based media by automatically generating SMIL code (Berke, 2004). MAGpie, freeware available from the National Center for Accessible Media, and Hi-Caption, a captioning tool for Flash, are typical of these products in that they require the manual entry of text and the manual determination of time codes (National Center for Accessible Media, n.d.; Yonaitis, 2003).

Commercially available software and web-based services are available to automate the process of determining time codes for captions. One method utilizes speech recognition to match data from an audio file and a manually generated text transcript (Aurix, 2003). A second method applies technology similar to automated speech
recognition, without attempting to perform actual recognition and employs linguistic skills to create and improve the algorithms that segment the text into appropriate captions. Speech processing engineers create and refine the audio synchronization technologies. The algorithms were internally developed and detailed technical data on the technology is not disclosed (K. Erler, personal communication, August 30, 2004; Automated Sync Technologies, 2004).

2.5 Structure of Speech

Speech is comprised of the articulation of words and intervening periods of silence. Words are composed of syllables, which in turn, are composed of segments or consonants and vowels. Words are combined to build phrases and sentences. Phrases and sentences comprise topics or paragraphs. Rhythm is the perception of time applied to phonetic events. This applies to how words and syllables are grouped in time and how segments make up syllables (Netsell, 1973). Speech rhythm functions mainly to organize the information bearing elements of speech into a coherent package (Allen, 1975).

Suprasegmental, or prosodic, features of speech are overlaid upon syllables, words, phrases, and sentences (Borden, Harris, & Raphael, 2003). These features include vocal frequency or pitch, stress or intensity, loudness, intonation, tempo, duration, juncture or pauses, and voice quality (Borden, Harris, & Raphael, 2003; Kent & Read, 1992; Netsell, 1973).

In order to describe and understand the overall temporal structure of speech, relative durations of phonemes (vowels, consonants, syllables), pauses within words, pauses between words, and pauses at ends of phrases, clauses and sentences must be taken into account (Zellner, 1994). These different categories of pauses may be
distinguished by their durations. Pauses between words are longer than those within words and pauses at the ends of phrases, clauses, and sentences are much longer than those within words or between words within a phrase (Kirsner, et. al., 2002; Zellner, 1994).

2.6 Pauses

Earlier research on pauses focused on the functions of pauses for speakers, such as breathing or formulating the next articulation, and the effects of cognitive variables, affective-state variables, and social interaction variables on the location and duration of pauses (Crystal, 1969; Rochester, 1973). Pauses were categorized according to content; silent or filled, if the pause consisted of the prolongation of a preceding sound or was filled with "um", "ah", or "er" (MacKay, 1978). Pauses were also categorized according to their location. Juncture or structural pauses occur at the end of speech segments and at syntactic positions and have syntactic function whereas hesitation pauses are not tied to linguistic events (Boomer & Dittman, 1962; Crystal, 1969). Filled pauses are much more likely to occur in conjunction with hesitation pauses than with juncture pauses (Boomer & Dittman, 1962). Low end thresholds were used in these early studies to define "significant" silent pauses, usually because of measurement limits or other experimental reasons. However, thresholds are arbitrary and cut out useful data (Campione & Veronis, 2002; Kirsner, Dunn, Hird, Parkin, & Clark, 2002).

More recent research on pauses has focused on the measurement and characterization of the number of and the duration of pauses in speech. Pause durations follow a log-normal distribution (Campione & Veronis, 2002; Kirsner, Dunn, Hird, Parkin, & Clark, 2002). In comparisons of read and conversational speech, the frequency
and duration of pauses were greater in conversational speech (Gustafson-Capkova & Megyesi, 2001, Walker, 1988). This difference was attributed to time for formulation of thought (Goldman-Eisler, 1972; Walker, 1998). Pauses within word groups get dropped when transcribed spontaneous speech is read later (Howell & Kadi-Hanifi, 1991). Silent pauses tend to occur at sentence boundaries for professional readers; phrase, clause and sentence boundaries for non-professional readers; and at turn and theme shift boundaries for dialogues (Gustafson-Capkova & Megyesi, 2001). In read speech, pauses corresponded to 92% of the punctuation marks inserted by transcribers; in spontaneous speech, pauses corresponded to 60% of placed punctuation (Guaitella & Santi, 1992).

Pause duration varies with prosodic and syntactic function. Previously reported values for pause duration for different syntactic functions are given in Table 1. Pauses at the end-of-sentences are longer than pauses within sentences and pauses at the ends of paragraphs or topic shifts are even longer (Fant & Kruckenberg, 1996; Goldman-Eisler, 1972; Guaitella & Santi, 1992; Howell & Kadi-Hanifi, 1991; Strangert, 1990; Swerts & Geluykens, 1994). In reading, pauses between complete sentences are longer and more uniform than pauses within sentences (Fant, Kruckenberg, & Ferreira, 2003; Goldman-Eisler, 1972).

These differences in pause duration are employed in modeling prosody for synthesized speech by placing long pauses at punctuation marks and placing shorter pauses according to lexical content (Emerard, Mortamet, & Cozanne, 1992; Zellner, 1998a; Zellner-Keller, 2002). In another approach to modeling synthesized prosody, pauses were placed to correspond to punctuation marks for read speech; for spontaneous speech, pauses were placed at random to simulate speaker hesitations (Guaitella & Santi,
1992). The performance of automatic syntactic parsing of transcribed speech is generally better when punctuation is included than when it is stripped out (Gregory, Johnson, & Charniak, 2004). In studies on automatically punctuating word streams, pause duration data were more useful than language model information (Kim & Woodland, 2003) and a combined acoustic-linguistic approach performed better than a linguistic only approach (Christensen, Gotoh, & Renals, 2001; Shriberg, Stolke, Hakkani-Tur, & Tur, 2000).

Table 1

Reported Values of Pause Duration for Various Syntactic Boundaries in Read Speech

<table>
<thead>
<tr>
<th>Reference</th>
<th>Phrases</th>
<th>Clauses</th>
<th>Sentences</th>
<th>Paragraphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Fant, Kruckenber, &amp; Ferreira, 2003)(^a)</td>
<td>360</td>
<td>530</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Fant, Kruckenber, &amp; Ferreira, 2003)(^b)</td>
<td>450</td>
<td>1100</td>
<td>1550</td>
<td></td>
</tr>
<tr>
<td>(Goldman-Eisler, 1972)(^c)</td>
<td>214</td>
<td>491</td>
<td>1199</td>
<td></td>
</tr>
<tr>
<td>(Strangert, 1990)(^a)</td>
<td>130</td>
<td>250</td>
<td>830</td>
<td>1475</td>
</tr>
<tr>
<td>(Swerts &amp; Geluykens, 1994)(^d)</td>
<td></td>
<td></td>
<td>1070</td>
<td>3810</td>
</tr>
</tbody>
</table>

Note: All values in ms.
\(^a\)News transcript. \(^b\)Novel excerpt. \(^c\)Radio show transcript. \(^d\)List of instructions.

2.7 Punctuation

Texts were originally analyzed with regard to how they would be spoken. Punctuation was used to indicate the rhythm and shape of a discourse. Even in prose, punctuation reflected the oratorical methods of order, connection and rhythm (Parkes, 1993). The indication of pauses, breathing spaces, pitch, and stress is still one of the
functions of punctuation. However, only a small fraction of modern writing is meant to be read aloud (Carey, 1958); and the role of punctuation has evolved beyond its original function (Nunberg, 1990). In the Renaissance, a shift began towards applying logical analysis to text. Consequently, punctuation began to serve to clarify the construction of written words, independent of reading aloud. In the nineteenth and twentieth centuries, punctuation and other graphic devices were used by novelists to indicate dialog and internal monologues (Parkes, 1993). Punctuation that simultaneously serves syntactic, semantic and prosodic functions, is strongly related to written syntax (Meyer, 1987), and has a linguistic sense that must be studied of its own accord (Nunberg, 1990). In most written text, the relationship between punctuation and prosody, especially within sentences, is weak and unsystematic; and therefore, only inconsistently marks prosodic boundaries (Sproat, 1998). However, in certain cases like transcripts of spoken discourse and verse, there is a greater degree of correspondence between punctuation marks and prosodic boundaries (Nunberg, 1990; Parkes, 1993). There are several conventions of syntactical punctuation that relate to timing. For example, single spaces correspond to the separation between words. Periods and colons correspond to relatively long pauses, semicolons correspond to moderately long pauses, and commas correspond to relatively short pauses (Complete Translation Services, Inc., 2004). There are also punctuation practices specific to captioning regarding the usage of quotation marks, hyphens and ellipses and how they relate to speech (Captioned Media Program, 2004). There is evidence that punctuation may be more useful than speech prosody markings in the syntactic parsing of transcribed speech for speech synthesis (Gregory, et al., 2004). Ends of sentences are denoted by periods, question marks, and exclamation points. However,
these punctuation points are sometimes ambiguous. A period can also denote a decimal point or an abbreviation. A period may simultaneously denote the end of a sentence and an abbreviation. Exclamation points and question marks can appear within quotation marks and parentheses. Ellipses (...) can occur both within sentences and at the ends of sentences, although an ellipsis at the end of a sentence should be followed by an end-of-sentence punctuation point (American Psychological Association, 2001). Previous observations of what percentage of periods in text denote abbreviations range from 7.8% in a corpus of scientific abstracts to 47% in the Wall Street Journal (Palmer and Hearst, 1997).

2.8 Silence Detection

There is an extensive body of literature on distinguishing between silence and speech. When creating an algorithm to detect transitions between silence and non-silence, it is desirable to use an adaptive, or self-normalizing solution that does not rely heavily on arbitrary fixed thresholds (deSouza, 1983) or on an a priori knowledge of the background noise (Savoji, 1989). There are five features of an audio signal that are generally used in silence detection (Atal, 1976). These are energy or magnitude, the zero crossing rate, the one sample delay autocorrelation coefficient, the linear predictive coding (LPC) predictor coefficient, and the LPC prediction error energy. Of these five features, signal energy or magnitude is the best for discerning between speech and silence (Aron, 1994). To achieve good performance a speech detector either has to employ multiple parameters or use a complex algorithm (Savoji, 1989). For example, a speech/silence detector based primarily on the magnitude of the signal is improved by using ZCR to distinguish between silence and relatively soft speech such as weak
fricatives (/l/, /th/, /h/), weak plosive bursts (/p/, /t/, /k/), nasal at the ends of words, and trailing vowel sounds at the ends of words (Rabiner and Schafer, 1978). Other parameters that can be used to supplement magnitude are a minimum continuous time to classify a signal as speech (Gan, 1988; Horii, 1983), and amplitude thresholds for determining transitions between silence and speech (Gan, 1988). The disadvantage of the minimum continuous time for classifying a signal as speech is that it is empirical or arbitrary. The threshold for transitions between speech and silence can be related to measures of the recording noise in an audio file. One method of measuring the recording noise involves calculating the RMS of the first 100 ms of a recording. This assumes that the recording begins with silence, which holds true for the recordings used in this study. However, if the assumption does not hold, it is possible to correct for this (Arons, 1994). Another way of determining the recording noise is to employ the observation (Hess, 1976) that histograms of energy levels tend to have peaks at levels corresponding to silence or recording noise. After determining this peak, Hess then set a silence threshold above this peak.

2.9 Summary

Pauses provide structure to speech by separating words into syntactic and prosodic groupings. In transcribed speech and read text, the location and duration of pauses correspond strongly to punctuation marks and to other linguistic features. This relationship has been utilized to model prosody for synthesized speech (Emerard, Mortamet, & Cozannet, 1992; Guaitella & Santi, 1992; Zellner, 1998a; Zellner-Keller, 2002; Gregory, Johnson, & Charniak, 2004) and to parse text streams into sentences and paragraphs (Christensen, Gotoh, & Renals, 2001; Shriberg, Stolke, Hakkani-Tur, & Tur,
2000; Kim & Woodland, 2003). However, no studies were found in the open literature that utilized this relationship to estimate caption time codes. Previous studies have found that the combination of text-based linguistic data and acoustically-based prosodic data yields more effective results than either data type used individually (Christensen, Gotoh, & Renals, 2001; Shriberg, Stolke, Hakkani-Tur, & Tur, 2000). This study will determine the applicability of combined acoustically-based data on pause location and duration and punctuation information from text to estimate time codes for captions. This study will also determine methods to incorporate the information for combining text and media and with captioning conventions and preferences, based upon the existing standards.
3.0 Overview

The purpose of this study was to develop methods and algorithms to effectively and efficiently determine time codes for captions from a plain-text transcript and a media file. Because previous studies have observed that combining text analysis with prosodic information is advantageous (Shriberg, Stolke, Hakkani-Tur, & Tur, 2000; Kim and Woodland, 2003), methods of combining information from audio analyses and transcript analyses will be employed. The general approach involved measuring pauses in speech by detecting silences in the audio data, parsing plain-text transcripts into sentences, utilizing observed trends between pause duration and textual features to estimate when sentence boundaries occur and generating the coding necessary to synchronize media with timed captions. The following sections will provide additional details.

3.1 Audio and Video Data

Several representative media were chosen in this study because of their high signal-to-noise ratios and their minimal background noise. Generally, transcripts or corresponding texts were either available or easily generated for the chosen media. The filenames, source, and duration of the media clips used in this study are shown in Table 2. Two radio program recordings (basraLibrary.wav and elections.wav) and their corresponding transcripts were obtained from National Public Radio (NPR). Only two segments from the two-hour long recordings were used in this study. These recordings were received as CD tracks, which were imported into iTunes and converted into WAV files using iTunes. Two other audio clips (letters3and4.wav and vollchl.wav) were mp3 files from a podcast of readings of Frankenstein (Booth Library, 2005). The mp3 files
were also imported and converted into WAV files using iTunes. WAV audio data was used because the data structure of this file format is well documented (Weber, n.d.), thereby making it possible to read the file as binary data and to calculate elapsed time.

Table 2 *Media Files Used in the Study.*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Source</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>basraLibrary.wav</td>
<td>NPR</td>
<td>4:21</td>
</tr>
<tr>
<td>elections.wav</td>
<td>NPR</td>
<td>3:43</td>
</tr>
<tr>
<td>letters3and4.wav</td>
<td>Booth Library Podcast</td>
<td>14:47</td>
</tr>
<tr>
<td>voll1ch1.wav</td>
<td>Booth Library Podcast</td>
<td>17:19</td>
</tr>
<tr>
<td>car.wav</td>
<td>WGBH</td>
<td>0:50</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>WGBH</td>
<td>1:20</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>WGBH</td>
<td>0:45</td>
</tr>
</tbody>
</table>

Three video clips were downloaded from the website of the PBS television program NOVA (WGBH Educational Foundation, 2005). The audio from these media clips were recorded in the WAV file format using Developer Edition 4.4 of Total Recorder on a PC with two Intel Pentium 4 3.06 GHz processors, 1.00 GB of RAM and 232 GB hard drive running on Microsoft Windows XP Professional Version Service Pack 2 operating system.

3.2 Parsing a Text Transcript into Sentences

Text transcripts were manually generated. Automatic speech recognition (ASR) has an error rate of 35 - 65% (Martone, Taskiran, & Delp, 2004; Shriberg, Stolke,
Estimation of Time Codes for Captioning

Hakkani-Tur, & Tur, 2000); consequently, the current state of the art in ASR technology is not sufficient to generate captions without human intervention. The NPR transcripts were received as plain text (.txt) files. These transcripts identified who was speaking each segment with the speakers last name followed by a colon. These identifiers were taken out manually using the WordPad text editor. The texts of the Frankenstein readings were obtained online as HTML documents from Frankenstein: The Pennsylvania Electronic Edition (Curran, n.d.). The text was copied and pasted into a WordPad document and saved as plain text to strip out any formatting characters. The transcripts for the videos from NOVA were manually generated by listening to the recordings and transcribing the spoken words.

Text transcripts were parsed into sentences because complete sentences are the preferred method of caption display (Kirkland, 1999). Transcripts of the media recordings were parsed into individual captions using a script written in PHP. PHP is an open source, cross-platform, general-purpose server-side scripting language that may be downloaded for free from the web. The text transcript was input as a plain text file. In the script, each character in the transcript was treated as an element of an array. Each array element was checked to see if it was a period, a question mark, or an exclamation point. If the character following an end-of-sentence punctuation mark was a space, a line-break, or a quotation mark, then that point was defined as a sentence boundary. The PHP script for parsing a transcript into sentences is shown in Appendix A.

3.3 Audio Data Analysis

Analyses of the audio data were performed only in the time domain. Pauses, or relative silence, were defined as segments below an amplitude threshold in the audio. It
was necessary to determine a threshold value for silence because the background noise of
the recordings was not absolutely zero. Durations and locations of pauses in recordings
were measured using scripts written in PHP. The resulting script read the format header
in the WAV files to determine sample rate, sample size, and total clip duration so that the
timing of pauses could be determined. The script then read the actual data from the WAV
files as binary data and detected periods of low amplitude. The PHP script for reading
the WAV files and detecting silences is given in Appendix B.

In order to manually measure the amplitude levels of recording noise and
background noise, the WAV audio files were viewed in Audacity, an open source audio
editing application. Portions of the file were played to identify audio features of interest.
The magnification was adjusted to achieve a good view of the portions of the audio file to
be measured.

A sample screen shot was taken of the waveform image, as illustrated in Figure
11. The waveform is typical in that segments of large amplitude, corresponding to
speech, are interspersed with segments of low amplitude, corresponding to pauses. These
segments of low amplitude vary in duration. The two end-of-sentence pauses have an
average duration of approximately 800 ms whereas pauses within sentences have an
average duration of about 200 ms. Of the end-of-sentence pauses; the pause beginning
around 1.7 seconds has an amplitude corresponding to the recording noise and the pause
beginning around 7.0 seconds has an amplitude corresponding to background noise, i.e.
breath intake or rustling papers. The amplitudes of areas identified as recording noise and
background noise were measured from the screenshot.
The audio waveforms were used to determine the locations and durations of pauses. The start time, end time, and durations of all pauses greater than 100 ms were tabulated. In order to test the relationship between pause duration and syntactic function, pauses were manually matched with the corresponding punctuation marks in the transcript. These manual measurements were used as a benchmark for the algorithms used to detect pauses and to estimate timecodes. As the file was played, the transcript was read. When a caption began, the audio player was paused and the position of the cursor was recorded. A PHP script was written to generate histograms of the amplitude data and to calculate the root mean square (RMS) of the amplitudes in the audio files. The sample interval used for the histogram was 20. The PHP code for the histogram and RMS calculation is given in Appendix C.

T-tests were performed for each audio sample to determine if the durations of pauses at ends of sentences are significantly greater than the durations of pauses within sentences. The overlap between the durations of end-of-sentence pauses and all other pauses greater than 100 ms was analyzed using the Mann-Whitney $U$ Test. Those two
statistical tests were employed to validate the assumption that all of end-of-sentence pause durations were greater than within-sentence pause durations. The pause durations were rank ordered and the ranks for the smaller sample were added together to determine the parameter U.

3.4 Web Interface

In order to facilitate data collection and algorithm testing, as well as enhance the usability of the automated captioning application, a web interface for running various scripts was developed using the PHP scripting language and a MySQL database. The web interface has the following functions:

1. To measure pauses;
2. To estimate caption timecodes automatically;
3. To determine timecodes manually;
4. To create SMIL, SAMI, and RealText files for captioning;
5. To enable downloading pause and timecode data.

The database associated with the captioning web interface consists of three tables: projects, pauses and timecodes. The data fields in each table are given in Table 3. The SQL scripts for creating the table are given in Appendix D.

The first step in using the web interface for captioning is to define a project. A web form is used to enter a project name; the media file name, the WAV audio file, the transcript file, the pause length threshold, and the amplitude threshold. Once a project is defined, the project can be selected from a dropdown menu of all defined projects. After selecting a project, a menu of functions becomes available. The first set of functions was
designed for scanning the audio, measuring pauses, and storing the pause data in the
database.

Table 3. Fields in Database Tables used in Web Interface.

Table: projects
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectID</td>
<td>Unique identifier for each project</td>
</tr>
<tr>
<td>projectName</td>
<td>Descriptive name for project</td>
</tr>
<tr>
<td>Media</td>
<td>Name of media file</td>
</tr>
<tr>
<td>Audio</td>
<td>Name of WAV audio file</td>
</tr>
<tr>
<td>Text</td>
<td>Name of text transcript file</td>
</tr>
<tr>
<td>timeThreshold</td>
<td>Minimum duration of pauses tabulated</td>
</tr>
<tr>
<td>amplitudeThreshold</td>
<td>Maximum amplitude of pauses</td>
</tr>
<tr>
<td>Cliptime</td>
<td>Total time of media/audio file</td>
</tr>
</tbody>
</table>

Table: pauses
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectID</td>
<td>Unique identifier for each project</td>
</tr>
<tr>
<td>pauseID</td>
<td>Pause index number (starts at 0)</td>
</tr>
<tr>
<td>startTime</td>
<td>Starting time of each pause (seconds)</td>
</tr>
<tr>
<td>endTime</td>
<td>Ending time of each pause (seconds)</td>
</tr>
<tr>
<td>Duration</td>
<td>Length of each pause (ms)</td>
</tr>
</tbody>
</table>

Table: timecodes
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectID</td>
<td>Unique identifier for each project</td>
</tr>
<tr>
<td>timecodeID</td>
<td>Timecode index number (starts at 0)</td>
</tr>
<tr>
<td>Manual</td>
<td>Manually determined timecode for a caption</td>
</tr>
<tr>
<td>closestPauseMatch</td>
<td>Pause closest to a manually determined timecode</td>
</tr>
<tr>
<td>Global</td>
<td>Timecode estimated from Global Threshold Method</td>
</tr>
<tr>
<td>localMax</td>
<td>Timecode estimated from Local Maxima Method</td>
</tr>
</tbody>
</table>
If pause data for a particular project is not found in the database, the available option is to scan the audio. Once the pause data is collected, the available option is to retrieve, view and download the pause data. After pause data has been collected, menu options for running the Global Threshold and Local Maxima timecode estimation algorithms become available. After these estimations are performed and the results stored in the database, the results can be viewed and downloaded for further analysis. The data downloads are in the form of plain text files and comma separated variable (CSV) files. A link is provided to the text file so that it can be saved in a local computer folder. Sample PHP scripting for creating a data file, reading data from a database, writing that data into the text file, and creating a link to the file is shown in Appendix E.

In order to provide a benchmark for the estimated caption timecodes, a web page was created to facilitate the manual determination of timecodes as shown in Figure 12. In the left hand frame, an instance of Windows Media Player was embedded in a web page with control buttons to Play, Pause and Stop the media player (Microsoft Corporation, 2004); and providing a slider control to move around within the media clip (Arvidsson, 2006). The original slider control coding had to be modified to interact properly with the media player. The coding to embed a media player and associated controls into a web page are given in Appendix F. JavaScript and information about the media player object were used to extract the current time from the media player and display that value in a text box labeled “Clip Time” as seen at the top of the right hand frame in Figure 12. Displayed concurrently with the media player were the captions for that media file. A submit button is associated with each caption segment and is set in a column labeled “Set Timecode.” When the submit button for a caption is pressed, the current media time is
submitted to and stored in the database. The timecode captured when a submit button is pressed is displayed in the column labeled “Manual timecode” as shown in Figure 12. The column labeled “Best Match” contains the results of a closest pause match. The duration of each pause is also displayed to the right of each caption text. The media player and the data capture pages were in one frame; data input buttons were placed in a separate web page to prevent the media player from reinitializing every time data was submitted. The coding to send manually determined timecodes to a database is given in Appendix G.

![Web Interface for the Manual Determination of Caption Timecodes](http://example.com/image.png)

**Figure 12.** Web Interface for the Manual Determination of Caption Timecodes.

### 3.5 Timecode Estimation Algorithms

During captioning, a text transcript file is split into caption segments at the ends of sentences. The goal of the timecode estimation is to determine the starting time and duration of each sentence (caption segment). Two timecode estimation algorithms were
developed in this study. Both of these algorithms generate initial estimates of the caption
timecodes based upon the analysis of the text transcript. In order to improve on the
accuracy of the time code estimates derived from the text analysis, timecodes estimated
from the text analysis were matched to pauses detected in the audio files.

3.5.1 Global Threshold Method

For the Global Threshold Method, three text analysis algorithms of increasing
complexity were developed and tested in this study. These algorithms are described
below.

3.5.1.1 Character Count Weighting (CCW) Algorithm

The Character Count Weighting (CCW) algorithm assumes that each character,
including blank spaces, in the text transcript corresponds to an equal amount of time.
This assumption is based on the observation that one of the factors in syllable timing is
the number of phonemes it includes (Campbell, 1992). The CCW algorithm starts by
determining the total clip time from the audio file. The text transcript file is then read and
split into captions at the ends of sentences. For every caption, the number of spaces, the
number of punctuation points within sentences, the number of punctuation points at the
ends of sentences, and the number of alphanumeric characters are tabulated.

The duration of each caption \( (t_{caption}) \) is estimated by calculating the ratio of the
number of characters in each caption \( (cc_{caption}) \) to the number of characters in the entire
transcript \( (cc_{total}) \) and multiplying that ratio by the clip time \( (t_{total}) \) as shown in Equation 1.

\[
t_{caption} = t_{total} \times \left( \frac{cc_{caption}}{cc_{total}} \right) \quad (1)
\]
An example of the Global Threshold Method text analyses is shown in Figure 13. The original text transcript is shown at the top of the figure. That is followed by the total clip time, the total word count and total character count for the transcript. The second section also shows the total number of punctuation and end punctuation points and the average duration assigned to those punctuation points in the text analyses. The last punctuation is not counted because it does not affect the timing estimates. The next section of the sample output shows how the captions are parsed into sentences and the count of spaces, within-sentence punctuation, end-of-sentence punctuation, and alphanumeric characters in each caption.

In the caption with index 2, for example, there are 14 spaces, one within-sentence punctuation point, one end-of-sentence punctuation point, and 73 alphanumeric characters for a total of 89 characters. The total number of characters for the entire transcript is 647 and the total time for the audio file is 44.404 s. The estimated duration for caption 2 is \( \frac{89}{647} \times 44.404 \) s, or 6.11 s.

3.5.1.2 Character Count Weighting and Punctuation Pauses (CCW + PP) Algorithm

To improve upon the accuracy of the previous method, the second algorithm retains the weighting by Character Count and adds factors to account for pauses corresponding to punctuation. The total duration of each caption \( t_{\text{caption}} \) is the sum of the duration of the pauses \( t_{\text{pauses}} \) and the duration of the speech segments \( t_{\text{speech}} \) in that caption as expressed in Equation 2.

\[
t_{\text{caption}} = t_{\text{pauses}} + t_{\text{speech}}
\]
Text File: car.avi.txt

Transcript: Someone watching a car accelerate toward light speed would see something very strange. It would seem as though the car itself were getting shorter and that time for the person in the car was slowing down. However, you wouldn't see these effects until the car began to approach the speed of light. At ninety percent of the speed of light, the car would appear to shrink to forty four percent of its usual length. This thought experiment answered Einstein's old question about what he would see if he traveled along with a beam of light. He simply couldn't make the trip, for at the speed of light, length would contract to zero and time would stop.

![Figure 13. Sample output of transcript parsing algorithm.](image-url)
In the CCW + PP algorithm, all characters, except for punctuation marks, are assumed to represent equal amounts of time. This algorithm has an additional assumption that the pauses with the longest durations correspond to punctuation marks in the transcript. The CCW + PP algorithm begins by determining the clip time \( t_{\text{total}} \) from the audio data. The amplitude data is then read from the audio file in order to detect regions of low amplitude. The start times, end times, and durations of all pauses greater than a defined pause duration threshold (100 ms) are tabulated. The text transcript file is then read and split into captions at the ends of sentences. The number of characters in each caption \( c_{\text{caption}} \) and in the entire transcript \( c_{\text{total}} \) are counted. The algorithm differentiates between punctuation points at the ends of sentences (periods, question marks, and exclamation points) and punctuation points within sentences (commas, semicolons, colons, and dashes).

The program then counts the number of end-of-sentence punctuation marks in each caption \( c_{\text{PES,caption}} \) and in the entire transcript \( c_{\text{PES,total}} \) as well as the number of punctuation points within sentences in each caption \( c_{\text{PWS,caption}} \) and in the entire transcript \( c_{\text{PWS,total}} \). Because the last end-of-sentence punctuation point corresponds to the end of the audio file, it is not used to estimate caption timing. The measured pauses are sorted by duration. The total \( \Sigma t_{\text{PES}} \) and average duration \( t_{\text{PES}} \) of the \( c_{\text{PES,total}} \) longest pauses are calculated. The average duration \( t_{\text{PES}} \) is assigned to end-of-sentence punctuation points. The total \( \Sigma t_{\text{PWS}} \) and average duration \( t_{\text{PWS}} \) of the next \( c_{\text{PWS,total}} \) longest pauses are calculated. This average \( t_{\text{PWS}} \) is assigned to within-sentence punctuation points. The total articulation time \( t_{\text{speech}} \) is estimated by subtracting the total punctuation time \( \Sigma t_{\text{PES}} + \Sigma t_{\text{PWS}} \) from the clip time \( t_{\text{total}} \). The number of non-
punctuation characters in each caption \( (cc_{non-punc,caption} = cc_{caption} - cc_{PES,caption} - cc_{PWS,caption}) \) and in the entire transcript \( (cc_{non-punc,total} = cc_{total} - cc_{PES,total} - cc_{PWS,total}) \) are calculated. The duration of each caption is estimated by calculating the ratio of the number of non-punctuation characters in each caption to the number of non-punctuation characters in the entire transcript and multiplying that ratio by the articulation time and then adding a factor to account for punctuation pauses by multiplying the number of punctuation marks in each caption by the average punctuation pause duration as shown in Equation 3.

\[
T_{caption} = (cc_{PES,caption} \times t_{PES}) + (cc_{PWS,caption} \times t_{PWS}) \\
+ (cc_{non-punc,caption} / cc_{non-punc,total}) \times (t_{total} - (\sum t_{PES} + \sum t_{PWS}))
\]  

(3)

where \( T_{caption} \) is the total duration of each caption, \( cc_{PES,caption} \) is the number of end-of-sentence punctuation marks in each caption, \( t_{PES} \) is the average duration of the end-of-sentence pauses, \( cc_{PWS,caption} \) is the number of within-sentence punctuation points in each caption, \( t_{PWS} \) is the average duration of within-sentence punctuation points, \( cc_{non-punc,caption} \) and \( cc_{non-punc,total} \) are the number of non-punctuation characters in each caption and the entire transcript, respectively, and \( t_{total} \) is the clip time, and \( (\sum t_{PES} + \sum t_{PWS}) \) is the total time assigned to punctuation pauses.

In the example shown in Figure 13, there are a total of four punctuation points within sentences and five end-of-sentence punctuation points that affect caption timing. The average duration of the five longest pauses is 1.792 s. The average duration of the four next longest pauses is 1.02 s. Again looking at the caption with index 2 in Figure 13, there are 14 spaces, one within-sentence punctuation point, one end-of-sentence
punctuation point, and 73 alphanumeric characters for a total of 87 non-punctuation characters. The total number of non-punctuation characters for the entire transcript is 638 and the total time not attributable to punctuation pauses is 31.364 s. The estimated duration for caption 2 is \((87/638) \times 31.364 \text{ s} + 1.792 \text{ s} + 1.02 \text{ s}\), or 7.09 s.

3.5.1.3 Character Count Weighting, Punctuation Pauses, and Pauses Between Words 

\((CCW + PP + PBW)\) Algorithm

The third algorithm not only retains the weighting by Character Count and punctuation, but also accounts for pauses between words. The additional assumption in this algorithm is that pauses between words are shorter than punctuation pauses and longer than pauses within words. The CCW + PP + PBW algorithm begins with the same procedure as the CCW + PP algorithm. It also counts the number of spaces between words not associated with punctuation points for the entire transcript \((cc_{spaces,\text{total}})\). After assigning the durations of the longest \((cc_{PES,\text{total}} + cc_{PWS,\text{total}})\) pauses to the end-of-sentence and within-sentence pauses, the total \((\Sigma t_{spaces})\) and average duration \((t_{spaces})\) of the next \(cc_{spaces}\) longest pauses are calculated and are associated with spaces between words without a punctuation point. The articulation time \((t_{speech})\) of the entire file is estimated by subtracting the total punctuation time \((\Sigma t_{PES} + \Sigma t_{PWS})\) and the total between-words pause time \((\Sigma t_{spaces})\) from the clip time \((t_{total})\). The duration of each caption is estimated by calculating the ratio of the number of alphanumeric characters in each caption

\[ cc_{\text{alphanumeric,caption}} = cc_{\text{caption}} - cc_{PES,\text{caption}} - cc_{PWS,\text{caption}} - cc_{spaces,\text{caption}} \]

and multiplying that ratio by the articulation time \((t_{speech})\), adding a factor to account for punctuation pauses by multiplying the number of punctuation marks
in each caption by the average punctuation pause duration, and adding an additional
factor to account for inter-lexical pauses by multiplying the number of spaces between
words in each caption by the average inter-lexical pause duration as shown in Equation 4.

\[ t_{\text{caption}} = (c_{\text{PES,caption}} \times t_{\text{PES}}) + (c_{\text{PWS,caption}} \times t_{\text{PWS}}) + (c_{\text{spaces,caption}} \times t_{\text{spaces}}) + (c_{\text{alphanumeric,caption}} / c_{\text{alphanumeric,total}}) \times t_{\text{speech,caption}} \]  

(4)

where \( t_{\text{caption}} \) is the total duration of each caption, \( c_{\text{PES,caption}} \) is the number of end-of-sentence punctuation marks in each caption, \( t_{\text{PES}} \) is the average duration of the end-of-sentence pauses, \( c_{\text{PWS,caption}} \) is the number of within-sentence punctuation points in each caption, \( t_{\text{PWS}} \) is the average duration of within-sentence punctuation points, \( c_{\text{spaces,caption}} \) is the number of spaces in a caption, \( t_{\text{spaces}} \) is the total between words pause time in a caption, \( c_{\text{alphanumeric,caption}} \) and \( c_{\text{alphanumeric,total}} \) are the number of alphanumeric characters in each caption and the entire transcript, respectively, and \( t_{\text{speech,caption}} \) is the total articulation time of a caption.

An example of the Global Threshold Method text analyses is shown in Figure 13. The original text transcript is shown at the top of the figure. That is followed by the total clip time, the total word count and total character count for the transcript. The second section also shows the total number of punctuation and end punctuation points (the last one is not counted because it does not affect the timing estimates) and the average duration assigned to those punctuation points in the text analyses. The next section of the sample output shows how the captions are parsed into sentences and the count of spaces,
within-sentence punctuation, end-of-sentence punctuation, and alphanumeric characters in each caption.

3.5.1.4 Feedback Using Temporal Proximity

The timecode estimations from the text analyses are based primarily on the number of characters (including spaces and punctuation) in each caption. However, there are other factors affecting the timing of speech (Campbell, 1992), including the fact that there are many words in English whose spellings are not phonetic. To further improve the caption accuracy, a feedback mechanism was developed. The process is described as follows:

1) The feedback algorithm starts with the caption start times estimated using the text analysis algorithms described above.

2) The caption start time estimated from the text analyses is matched to the end time of the closest pause detected in the audio data. The purpose of this step is to correct for small errors in the caption durations estimated from the text analysis. Each pause end time is used only once so that multiple captions are not assigned the same timecode.

3) Calculate the differences between caption durations estimated from the text analysis and from the matching pause locations detected in the audio file.

4) Starting at the beginning of the audio, the estimated caption start times are then shifted to correspond to neighboring pause end times to minimize the differences in caption duration calculated from the text analysis and audio matching. This step is performed in order to correct for any large-scale errors resulting from Step 2.

After each caption start time shift, all caption durations are recalculated and steps 3 and 4 are repeated until the last caption is tested. When using only a temporal
proximity criterion for selecting pauses, all detected pauses, regardless of their duration, are used. An example of the caption times resulting from the Global Threshold Method text analyses and from the Temporal Proximity Feedback is shown in Figure 14.

<table>
<thead>
<tr>
<th>Text File: car.avi.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCW</td>
</tr>
<tr>
<td>2.327</td>
</tr>
<tr>
<td>5.948</td>
</tr>
<tr>
<td>20.335</td>
</tr>
<tr>
<td>28.220</td>
</tr>
<tr>
<td>36.727</td>
</tr>
</tbody>
</table>

*Figure 14.* Sample output of caption times estimated from the text analyses and the Temporal Proximity Feedback algorithm from the Global Threshold Method.

Tests were run to see if the accuracy of the feedback method can be improved by limiting the pauses for matching to those likely to correspond to the ends of sentences and clauses, based on the correlation of pause duration data and pause function. The premise is that if a relatively short pause is near the estimated caption time, it is ignored in favor of longer pauses that are more likely to be associated with the end of a sentence. In this algorithm, the detected pauses are sorted by duration from longest to shortest. The pool of pauses for temporal proximity matching is determined by the threshold factor. The minimum number of pauses in the pool is the number of captions \( n \). This number is multiplied by a threshold factor to set the number of pauses in the pool of likely pauses.
3.5.2 Local Maxima Method

In the Local Maxima Method, the text is parsed into captions using end-of-sentence punctuation as break points. The number of end-of-sentence punctuation points \( n \) equals the number of captions. A total articulation time is estimated by subtracting the total duration of the \( n \) longest pauses from the total cliptime. The duration of each caption articulation time is estimated with Character Count Weighting; that is dividing the number of characters in each caption by the total number of characters in the transcript.

The estimated timeline of the captions is then constructed by:

1) moving out caption duration \( (d_1) \)
2) querying pauses in an area centered on \( d_1 \)
3) using a range that is defined as a percentage of the estimated duration of the preceding caption with a set minimum range
4) finding the longest pause starting within that range
5) adding the chosen pause duration to the estimated caption duration
6) moving out caption duration \( (d_2) \)
7) and then repeating the process to the end of the transcript

The Local Maxima Method constructs a timeline without using average values for pauses at ends of sentences, but rather uses actual pause duration values. When matching estimated times from text analysis to the location of pauses measured from the audio file, the local maxima method limits the pause pool by limiting the time range searched; whereas the global threshold method limits the pause pool by a pause duration criterion.

An example of the Local Maxima Method text analyses is shown in Figure 15.

The total clip time, the total word count and total character count for the transcript is
<table>
<thead>
<tr>
<th>Caption Index</th>
<th>Caption Start Time Estimated from CCW Text Analysis (s)</th>
<th>Caption Duration Estimated from Text Analysis (s)</th>
<th>Lower Bound (s)</th>
<th>Upper Bound (s)</th>
<th>Pause with Local Maximum Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.327</td>
<td>4.785</td>
<td>----</td>
<td>----</td>
<td>2.327</td>
</tr>
<tr>
<td>1</td>
<td>7.112</td>
<td>6.510</td>
<td>5.612</td>
<td>8.612</td>
<td>7.610</td>
</tr>
<tr>
<td>2</td>
<td>14.120</td>
<td>5.063</td>
<td>12.493</td>
<td>15.748</td>
<td>15.016</td>
</tr>
<tr>
<td>3</td>
<td>20.079</td>
<td>6.343</td>
<td>18.579</td>
<td>21.579</td>
<td>21.014</td>
</tr>
<tr>
<td>4</td>
<td>27.357</td>
<td>6.844</td>
<td>25.371</td>
<td>29.343</td>
<td>29.333</td>
</tr>
<tr>
<td>5</td>
<td>35.676</td>
<td>6.176</td>
<td>34.466</td>
<td>37.888</td>
<td>36.976</td>
</tr>
</tbody>
</table>

*Figure 15. Sample output of transcript parsing and local maxima analysis.*

shown at the top, followed by the total number of end punctuation points (the last one is not counted because it does not affect the timing estimates) and the total duration assigned to the end punctuation points in the text analyses. The captions are parsed into sentences and the spaces, within-sentence punctuation, end-of-sentence punctuation, and alpha–numeric characters in each caption are counted as in the previous example.

The final section of the sample output shows the estimated caption time calculated from the start time of the previous caption and the estimated duration of the previous caption as calculated from the Character Count Weighting (CCW) and the articulation time, which is the Media duration minus the total duration of the five longest pauses. The lower and upper bounds are the limits of the search for the pause with maximum duration. The local maxima column gives the end point of the pause with maximum duration within the search range.
3.5.2.1 Automatic Correction Algorithm

An automatic correction algorithm was added to the local maxima method to test whether accuracy could be improved. If a transcript is split into \( n \) captions, a duration threshold of pauses most likely to correspond to ends of sentences is set as the duration of the \( n^{th} \) longest pause detected in the audio file. When the longest pause for any caption end is chosen, all other pauses within the search range that have durations greater than or equal the threshold of likely pauses are also selected from the pause data. As this data is collected, a few testing parameters are calculated:

1) Total number of local maxima determined
2) Total number of local maxima whose durations are greater than the threshold of likely pauses
3) The difference between estimated caption duration from the text algorithm and the duration resulting from choosing a particular pause (\( \Delta t \)).

The correction algorithm is as follows:

1) If the local maximum is less than the likely pause threshold, choose that maximum and add one to the number of maxima determined.
2) If the local maximum is greater than the likely pause threshold and no other pauses in the search range are greater than the likely pause threshold, choose that maximum, add one to the number of maxima determined and add one to the number of local maxima greater than the likely pause threshold.
3) If the local maximum is greater than the likely pause threshold and there are other pauses greater than the likely pause threshold in the search range, choose the local maximum for now and calculate \( \Delta t \) for the local maximum and the other
pauses greater than the likely threshold. If the local maximum and the minimum $\Delta t$ agree, choose the local maximum and add one to the hits and tries. If the local maximum pause and the minimum $\Delta t$ disagree, choose the local maximum for now, set the pause giving the minimum $\Delta t$ as the alternative choice, add one to the hits and tries, and set a checkpoint at the caption. After a caption estimation timeline is completed for a media clip, the checkpoints are compiled. A second run through the timeline is made using the alternative choices at the checkpoints to see if the number of hits increases. If the number of hits increases, the alternative choice is used. If the number of hits is not increased, the choice made in the first run is retained.

3.5.2.2 Manual Reset Algorithm

A manual reset algorithm was added to the local maxima method to provide a user-controlled means to further improve the caption accuracy. When a transcript is split into $n$ captions, a duration threshold of pauses most likely to correspond to ends of sentences is set as the duration of the $n^{th}$ longest pause detected in the audio file. When the longest pause for any caption end is chosen, all other pauses within the search range that have durations greater than or equal the threshold of likely pauses are also selected from the pause data. As this data is collected, two testing parameters are calculated:

1) Total number of local maxima determined

2) Total number of local maxima whose durations are greater than the threshold of likely pauses

The manual reset algorithm works as follows:
1) Determine local maximum pause.

2) Update testing parameters. If the local maximum is less than the likely pause threshold, choose that maximum and add one to the number of maxima determined. If the local maximum is greater than the likely pause threshold and no other pauses in the search range are greater than the likely pause threshold, choose that maximum, add one to the number of maxima determined and add one to the number of local maxima greater than the likely pause threshold.

3) Whenever three local maxima that are below the likely pause threshold are accumulated, manual input is requested.

4) If the local maximum is greater than the likely pause threshold and there are other pauses greater than the likely pause threshold in the search range, manual input is requested.

5) After a manual input is performed, the number of tries and the number of hits are both reset to zero and the local maxima method is resumed.

When a manual reset is triggered, a link to a modified manual timecode determination interface is generated. When the user clicks on the link, the web interface opens in a new browser window. The media player is started and is fast-forwarded to 15 seconds before the caption time estimated from the text analysis. The time input form is only for the current caption.

3.6 Effect of Amplitude Threshold on Timecode Estimation Error

In order to understand the effects of amplitude threshold on the timecode estimation and to optimize the use of threshold, tests were conducted with varying values for the amplitude threshold. Both the Global Threshold and the Local Maxima methods
were investigated. In the study, the average error, defined as the difference between the actual and estimated timecodes, was calculated for each run. The global RMS was calculated from the audio data file, whereas the recording noise and background noise were measured from the plotted waveforms.

3.7 Data Analysis

The estimated caption times generated by the text analysis and feedback algorithms were analyzed by calculating the difference between the estimated caption times and the actual caption times for each media file. The mean and standard deviation of these differences for each algorithm are calculated for each media file. The above-mentioned algorithms were compared in terms of the mean difference, in order to identify the best algorithm for caption timecode estimate.

3.8 Development of Code to Generate Captioned Media

After the transcripts were parsed into captions at the ends of sentences and the caption timecodes were estimated using the preceding analyses, they were stored in a database. The next step was to utilize the estimation results by putting them into a form that actually synchronizes text captions and media for viewing. These data were retrieved and inserted into templates for each media synchronization format. The resulting text was saved as a plain text file using server-side scripting. The following media players were used in this study:

- RealPlayer 10.5
- QuickTime 7.1.3
- Windows Media Player 10
- Macromedia Flash Player 8
The formats for synchronizing text captions with audio or video provided by the code in this study are:

- SMIL for Real Player,
- SMIL for QuickTime,
- SAMI for Windows Media Player
- XML captions for Flash

Examples of these text-based formats are shown in Chapter 2, Section 3. The PHP code for generating these formats is given in Appendix H.

3.9 Summary

This chapter describes the algorithms and methods used to automate the process of synchronizing text captions to audio or video. Audio files were converted to the WAV format and the audio of video files were recorded as WAV files. Once converted to the WAV format, the audio data was analyzed to determine regions of speech and silence. The starting times, ending times and durations of the detected pauses were stored in a database. The text transcripts of the media clips were parsed into sentences using end-of-sentence punctuation points in the transcripts. The number of characters in each caption was counted. The characters in each caption were categorized as a punctuation mark, a space, or an alphanumeric. The punctuation marks and spaces were assigned durations based on the durations of pauses measured in the audio data. The alphanumeric characters in each caption were assigned durations as a proportion of the total articulation time of an audio file. Caption durations were estimated by adding these character durations together.

Two methods for correlating ends of sentences to pauses in the audio data were developed. Both methods utilized textual and audio data. However, the Global Threshold
Method relied on average global values of pauses to estimate the timing of captions while the Local Maxima Method used information about individual pauses in specific segments of the audio data. Methods of further incorporating the audio data as a feedback mechanism were also evaluated. A web interface was developed to facilitate the implementation of the automated methods, the manual determination of caption times, and the collection of data for analysis. The estimated timecodes were incorporated into scripts that automatically generated the XML documents to synchronize the media and text captions in RealPlayer, QuickTime, Windows Media Player, or Flash Player.
4.1 Parsing of Text Transcripts

Text transcripts were parsed into sentences because complete sentences are the preferred method of caption display (Kirkland, 1999) and because sentence boundaries potentially correspond to detectable audio phenomena. There were a total of 341 sentences in the seven text transcripts used in this study. The algorithm, which looked for an end-of-sentence punctuation mark and a following space, correctly identified 338 punctuation points as sentence boundaries and correctly identified two periods used to denote the initials in a name as not denoting sentence boundaries. There were three (3) cases of ambiguous punctuation points: one exclamation point in the middle of a sentence and two cases of title abbreviations, e.g. Mrs. The algorithm incorrectly identified these as sentence boundaries, resulting in an overall error rate of 0.9%.

Because there were a small number of ambiguous punctuation points, they were manually removed from the transcripts. However, this is not a viable option in the long term. Previous studies on Sentence Boundary Detection (SBD) have also noted the issue of ambiguous punctuation marks and its negative effect on properly identifying sentence boundaries (Palmer and Hearst, 1997; Kiss & Strunk, 2002; Mikheev, 2000). The measures taken to solve this problem include maintaining a list of abbreviations for comparing observed patterns of space-letter(s)-period in a text (Munoz and Nagarajan, 2001) or employing pattern recognition and statistical modeling (Palmer and Hearst, 1997; Kiss & Strunk, 2002; Mikheev, 2000). In order to train systems for punctuation mark disambiguation, samples of text around punctuation points need to be tagged.
Estimation of Time Codes for Captioning

58

according to their part-of-speech (Palmer & Hearst, 1997; Mikheev, 2000). The function
of many SBD algorithms is to parse existing text corpora such as electronic versions of
the Wall Street Journal. In this study, four of the seven transcripts were either pre-
existing text from a book or transcribed by an external source. The text from a book was
in paragraph form and the transcripts from NPR were split into paragraphs based on a
change in speaker.

Based on the above observations, the transcripts generated for this study were also
done in paragraph format. However, in practice, users will create their own transcripts
and the text formatting can be adapted to the needs of the captioning process. Most
captioning style guides suggest that punctuation in captioning “follow the conventional
rules for punctuation and spelling as outlined in standard English-language style manuals
and dictionaries” (Media Access Group, 2002). However, the transcription of certain
speech constructions sometimes requires use of punctuation that is unique to the
captioning process (Captioned Media Program, 2004). For instance, Caption Colorado, a
major captioning provider, instructs its new trainees to insert a new line-formatting
symbol after all end-of-sentence punctuation marks (Caption Colorado, 2005). If a
transcriber is instructed to insert line-breaks after all sentence ending punctuation marks,
then the algorithm for identifying ends of sentences in a transcript could search for the
pattern [end-of-sentence punctuation – line break] and the problem of ambiguous
punctuation would be solved. Similar formatting is used to separate blocks of caption
text when importing text from Notepad into MAGpie captioning software (National
Center for Accessible Media, n.d.-b).
4.2 Audio Data Analysis

The audio data were analyzed by observing the distributions of all pauses greater than 100 ms in each media file. The pauses that corresponded to the ends of sentences were identified by listening to the audio files using Audacity, pausing the player at points corresponding to punctuation marks in the transcript and manually measuring the location and duration of the corresponding pause. Pause duration (y-coordinate) was graphed as a function of time (x-coordinate).

Figure 16 illustrates the pause duration as a function of location in the media timeline for the audio file “car.wav.” The open circles represent the within-sentence pauses greater than 100 ms and the solid squares represent the end-of-sentence pauses. The end-of-sentence pauses range in duration from 831 to 2232 ms. The maximum duration of the within-sentence pauses is 739 ms. The longer within-sentence pauses (greater than 250 ms) occur at punctuation points such as commas and semi-colons as well as at other phrase and clause boundaries that occur at function words such as "and", "but", and "or" (Goldman-Eisler, 1972). The shorter within-sentence pauses (100 ms – 250 ms) correspond to longer breaks between individual words. Pauses within words, i.e. between syllables, are in the range of 10 ms – 50 ms are therefore not shown in this graph. This file represents an “ideal” case in which the durations of the end-of-sentence pauses are all greater than those of the within-sentence pauses.
The file basraLibrary.wav (Figure 17) represents the other extreme. Although the durations of the pauses at the ends of sentences appear to be on average much greater than the durations of other pauses, there is a great deal of overlap between the durations of the end-of-sentence pauses and other pauses and the range of the end-of-sentence pause durations are high. In this case, the durations of the end-of-sentence pauses range from 142 to 2026 ms. The maximum duration of the within-sentence pauses is 1016 ms and there are 17 more within pauses ranging from 500 – 1000 ms in duration. The three lowest end-of-sentence pauses (150 – 200 ms) were identified as transitions between two speakers that perhaps could have been represented by within-sentence punctuation.
In order to quantify these observations, statistical analyses were conducted. T-tests were performed to determine if the durations of the end-of-sentence pauses are significantly greater than the durations of the within-sentence pauses. The variance in the durations of the end-of-sentence pauses in each recording was quantified by standard deviations. The durations of the within-sentence pauses and end-of-sentence pauses are presented in Table 4. For two of the files, the difference was significant at the .05 level. For the other five files, the difference was significant at the .001 level. In summary, there is a significant difference between the durations of end-of-sentence pauses and within-sentence pauses.
Table 4

*Differences in Durations of Pauses Corresponding to Within-Sentence Punctuation and End-of-Sentence Punctuation.*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Within-Sentence</th>
<th>End-of-Sentence</th>
<th>df</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td>basraLibrary.wav</td>
<td>415.20</td>
<td>200.14</td>
<td>844.56</td>
<td>413.10</td>
</tr>
<tr>
<td>car.wav</td>
<td>614.47</td>
<td>93.45</td>
<td>1282.36</td>
<td>581.95</td>
</tr>
<tr>
<td>elections.wav</td>
<td>536.48</td>
<td>249.39</td>
<td>1021.26</td>
<td>318.21</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>457.94</td>
<td>121.88</td>
<td>3860.01</td>
<td>3438.03</td>
</tr>
<tr>
<td>letters3and4.wav</td>
<td>458.72</td>
<td>210.75</td>
<td>1176.50</td>
<td>479.89</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>554.78</td>
<td>242.90</td>
<td>9297.78</td>
<td>1053.48</td>
</tr>
<tr>
<td>vollchl.wav</td>
<td>451.26</td>
<td>149.50</td>
<td>808.71</td>
<td>263.58</td>
</tr>
</tbody>
</table>

*p < .05, one-tailed. **p < .001, one-tailed.

The degree of overlap between the durations of end-of-sentence pauses and the within-sentence pauses in each recording was determined by calculating the Mann-Whitney U parameter. In order to facilitate comparison between recordings, the standard deviation was normalized by dividing each standard deviation by the corresponding mean to get the coefficient of variation and the Mann-Whitney U parameter was normalized by dividing it by the total number of pauses in a recording (n). These results are given in Table 5.
Table 5

*Mann-Whitney U Test on Pause Duration Normalized by Total Number of Pauses Associated with Punctuation Points.*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Within-Sentence Pauses ($n_{ws}$)</th>
<th>End-of-Sentence Pauses ($n_{eos}$)</th>
<th>$U$</th>
<th>$U/(n_{ws} + n_{eos})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>basraLibrary.wav</td>
<td>53</td>
<td>42</td>
<td>359.5</td>
<td>3.78</td>
</tr>
<tr>
<td>car.wav</td>
<td>5</td>
<td>4</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>elections.wav</td>
<td>60</td>
<td>39</td>
<td>270.5</td>
<td>2.73</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>6</td>
<td>8</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>letters3and4.wav</td>
<td>265</td>
<td>119</td>
<td>1304.0</td>
<td>3.40</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>6</td>
<td>2</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>vollchl.wav</td>
<td>258</td>
<td>118</td>
<td>2894.5</td>
<td>7.70</td>
</tr>
</tbody>
</table>

Generally, the larger $U$ values represent more overlap between the variables.

Three of the files (car.wav, elevator.wav, and spacetime.wav) had no overlap (with zero $U$ value) in the durations of end-of-sentence pauses and within-sentences; the other four files had varying levels of overlap in the durations of the two types of pauses. The observed relationships between pause duration and syntactic function in this study are consistent with previous studies, with pauses between sentences significantly longer than pauses within sentences (Fant, Kruckenburg, Ferriera, 2003; Goldman-Eisler, 1972; Strangert, 1990; Swerts and Geluykens, 1994) but with overlap in the durations of the two groups (Goldman-Eisler, 1972; Swerts and Geluykens, 1994). This may indicate that
using absolute global values for pause duration for different classes of pauses may be inaccurate because of variations in the speech rate over the course of a recording.

However, within localized regions of recordings, the relationship between pause duration and syntactic function appears to be stronger than over the entire file, as seen in Figure 17 in the ranges of 0 to 70 seconds, 70 to 170 seconds, and 170 seconds to the end of the recording. In order to confirm this observation, the four audio files with non-zero Mann-Whitney parameters were split into segments at points where there were observable changes in the pause durations. T-tests were performed and Mann-Whitney parameters were calculated for each of the time segments. The durations of the within-sentence pauses and the end-of-sentence pauses within segments of the non-ideal audio files are given in Table 6. The four files were split into 21 segments. Seventeen of the twenty-one segments retained a significant difference between the durations of within-sentence and end-of-sentence pause durations. Eighteen of twenty-one segments had lower standard deviations in the durations of end-of-sentence pauses than their corresponding files. The normalized Mann-Whitney U parameters for the segments of the audio files are given in Table 7. All twenty-one segments had lower levels of overlap in the durations of the two types of pauses than the audio files taken as a whole. These results indicate that in terms of successfully distinguishing between within-sentence pauses and end-of-sentence pauses on the basis of duration, it is advantageous to analyze data over localized portions of a file rather than over the entire audio file.
Table 6

*Differences in Durations of Pauses Corresponding to Within-Sentence Punctuation and End-of-Sentence Punctuation for Segments of Non-ideal Audio Files.*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Within-Sentence</th>
<th>End-of-Sentence</th>
<th>df</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td>basraLibrary.wav</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 – 30 s</td>
<td>234.73</td>
<td>87.09</td>
<td>520.07</td>
<td>101.39</td>
</tr>
<tr>
<td>30 – 70 s</td>
<td>513.99</td>
<td>189.57</td>
<td>607.38</td>
<td>12.44</td>
</tr>
<tr>
<td>70 – 140 s</td>
<td>439.83</td>
<td>286.40</td>
<td>1083.26</td>
<td>389.96</td>
</tr>
<tr>
<td>140 – 142 s</td>
<td>152.20</td>
<td>41.97</td>
<td>122.20</td>
<td>--</td>
</tr>
<tr>
<td>142 – 195 s</td>
<td>431.20</td>
<td>186.57</td>
<td>763.29</td>
<td>243.72</td>
</tr>
<tr>
<td>195 – 205 s</td>
<td>165.11</td>
<td>96.69</td>
<td>132.01</td>
<td>9.77</td>
</tr>
<tr>
<td>205 – 260 s</td>
<td>329.52</td>
<td>136.86</td>
<td>1043.42</td>
<td>374.74</td>
</tr>
<tr>
<td>elections.wav</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 – 30 s</td>
<td>171.80</td>
<td>88.48</td>
<td>837.22</td>
<td>278.96</td>
</tr>
<tr>
<td>30 – 90 s</td>
<td>450.89</td>
<td>346.61</td>
<td>991.74</td>
<td>221.30</td>
</tr>
<tr>
<td>90 – 150 s</td>
<td>678.90</td>
<td>191.31</td>
<td>1257.84</td>
<td>225.98</td>
</tr>
<tr>
<td>150 – 200 s</td>
<td>514.40</td>
<td>232.90</td>
<td>1200.21</td>
<td>221.26</td>
</tr>
<tr>
<td>200 – 223 s</td>
<td>587.38</td>
<td>268.69</td>
<td>794.02</td>
<td>309.06</td>
</tr>
<tr>
<td>letters3and4.wav</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 – 200 s</td>
<td>473.92</td>
<td>200.62</td>
<td>1137.97</td>
<td>671.46</td>
</tr>
<tr>
<td>200 – 550 s</td>
<td>394.82</td>
<td>175.42</td>
<td>1230.30</td>
<td>380.19</td>
</tr>
<tr>
<td>550 – 886 s</td>
<td>506.56</td>
<td>227.60</td>
<td>1133.98</td>
<td>434.55</td>
</tr>
<tr>
<td>vollch1.wav</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 – 100 s</td>
<td>385.55</td>
<td>105.76</td>
<td>814.11</td>
<td>78.16</td>
</tr>
<tr>
<td>100 – 270 s</td>
<td>469.26</td>
<td>142.43</td>
<td>831.76</td>
<td>308.63</td>
</tr>
<tr>
<td>270 – 490 s</td>
<td>390.73</td>
<td>114.38</td>
<td>791.20</td>
<td>53.46</td>
</tr>
<tr>
<td>490 – 650 s</td>
<td>434.67</td>
<td>185.44</td>
<td>878.00</td>
<td>355.11</td>
</tr>
<tr>
<td>650 – 850 s</td>
<td>580.14</td>
<td>182.17</td>
<td>882.58</td>
<td>226.94</td>
</tr>
<tr>
<td>850 – 1040 s</td>
<td>416.55</td>
<td>149.08</td>
<td>716.85</td>
<td>171.04</td>
</tr>
</tbody>
</table>

* p < .001, one-tailed.
Table 7

*Mann-Whitney U Test on Pause Duration Normalized by Total Number of Pauses Associated with Punctuation Points for Segments of Non-ideal Audio Files.*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Within-Sentence Pauses (n_{ws})</th>
<th>End-of-Sentence Pauses (n_{eos})</th>
<th>U</th>
<th>U/(n_{ws} + n_{eos})</th>
</tr>
</thead>
<tbody>
<tr>
<td>basraLibrary.wav</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 - 30 s</td>
<td>7</td>
<td>5</td>
<td>1.0</td>
<td>0.08</td>
</tr>
<tr>
<td>30 - 70 s</td>
<td>10</td>
<td>3</td>
<td>5.5</td>
<td>0.42</td>
</tr>
<tr>
<td>70 - 140 s</td>
<td>12</td>
<td>14</td>
<td>13.0</td>
<td>0.50</td>
</tr>
<tr>
<td>140 - 142 s</td>
<td>3</td>
<td>1</td>
<td>1.0</td>
<td>0.25</td>
</tr>
<tr>
<td>142 - 195 s</td>
<td>15</td>
<td>8</td>
<td>15.5</td>
<td>0.67</td>
</tr>
<tr>
<td>195 - 205 s</td>
<td>12</td>
<td>2</td>
<td>10.5</td>
<td>0.75</td>
</tr>
<tr>
<td>205 - 260 s</td>
<td>11</td>
<td>9</td>
<td>4.0</td>
<td>0.20</td>
</tr>
<tr>
<td>elections.wav</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 - 30 s</td>
<td>8</td>
<td>9</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>30 - 90 s</td>
<td>15</td>
<td>12</td>
<td>12.0</td>
<td>0.44</td>
</tr>
<tr>
<td>90 - 150 s</td>
<td>13</td>
<td>7</td>
<td>2.0</td>
<td>0.10</td>
</tr>
<tr>
<td>150 - 200 s</td>
<td>18</td>
<td>9</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>200 - 223 s</td>
<td>6</td>
<td>2</td>
<td>2.0</td>
<td>0.25</td>
</tr>
<tr>
<td>letters3and4.wav</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 - 200 s</td>
<td>66</td>
<td>28</td>
<td>120.0</td>
<td>1.28</td>
</tr>
<tr>
<td>200 - 550 s</td>
<td>105</td>
<td>47</td>
<td>36.0</td>
<td>0.24</td>
</tr>
<tr>
<td>550 - 886 s</td>
<td>94</td>
<td>45</td>
<td>305.5</td>
<td>2.20</td>
</tr>
<tr>
<td>vollch1.wav</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 - 100 s</td>
<td>23</td>
<td>12</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>100 - 270 s</td>
<td>40</td>
<td>20</td>
<td>113.0</td>
<td>1.88</td>
</tr>
<tr>
<td>270 - 490 s</td>
<td>62</td>
<td>31</td>
<td>126.0</td>
<td>1.36</td>
</tr>
<tr>
<td>490 - 650 s</td>
<td>52</td>
<td>15</td>
<td>85.0</td>
<td>1.27</td>
</tr>
<tr>
<td>650 - 850 s</td>
<td>33</td>
<td>16</td>
<td>78.0</td>
<td>1.59</td>
</tr>
<tr>
<td>850 - 1040 s</td>
<td>48</td>
<td>24</td>
<td>90.0</td>
<td>1.25</td>
</tr>
</tbody>
</table>

* p < .001, one-tailed.
4.3 Global Threshold Method

The Global Threshold Method consists of estimating caption durations using character count weighting, refining those estimates by assigning average pause durations that correspond to punctuation marks, and correcting the estimates by shifting the start times of captions to the location of the nearest pause detected in the audio recording. Because of errors in the text-based estimations, the nearest pause may be a pause that does not correspond to a punctuation point. Because pauses corresponding to punctuation marks in the text tend to be the longest in duration, an attempt was made to improve the accuracy of the pause matching by using only the pauses that are longest in duration. A pause duration threshold factor is used to regulate the number of pauses in the pool for closest pause matching. However, in order to account for overlap in the duration between punctuation pauses and other measured pauses, the number of pauses had to be greater than the number of punctuation pauses.

In order to optimize this parameter, the effect of the pause duration threshold parameter on the accuracy of the global threshold algorithm was tested for the files vollchl.wav at an amplitude threshold of 5000 and elections.wav at an amplitude threshold of 1000. The results of these tests are shown in Figure 18. The Average Error in Timecode estimation (y-coordinate) is plotted as a function of the Pause Duration Threshold Factor used (x-coordinate). For both cases, the maximum error occurred at a Pause Duration Threshold Factor of 1.0. As the threshold factor was increased to around 1.5, the Average Error decreased. As the Pause Duration Threshold Factor was increased further, the average error did not change. Based on these observations, limiting the pauses available to choose from does not improve the accuracy of the timecode
estimation algorithm. Once the pause duration threshold factor reaches a certain point (about 1.5), the error magnitude reaches a minimum value and does not decrease any further. This value for the threshold factor was used in subsequent runs of the global threshold method.

![Effect of Pause Duration Threshold Factor on Average Error Magnitude](image)

*Figure 18. Effect of Pause Duration Threshold Factor on Average Error Magnitude*

The results of the text analyses and the closest pause match for the global threshold method are given in Table 8. For a few of the files; namely car.wav, spacetime.wav, and elevator.wav; the accuracy of the analyses improves as the punctuation and inter-lexical pauses are accounted for as the closest pause match is performed. For the other files, the average error magnitude is not improved and actually increases. In order to better understand these results it is necessary to relate them to the characteristics of the pause durations of each recording.
Table 8

*Average Error Magnitude for Text Analysis Algorithms and global pause matching.*

<table>
<thead>
<tr>
<th>Filename</th>
<th>CCW (s)</th>
<th>CCW+PP (s)</th>
<th>CCW+PP+IP (s)</th>
<th>Global Closest Pause Match (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>basraLib.wav</td>
<td>2.833</td>
<td>2.007</td>
<td>2.704</td>
<td>2.989</td>
</tr>
<tr>
<td>car.wav</td>
<td>0.734</td>
<td>0.542</td>
<td>0.597</td>
<td>0.000</td>
</tr>
<tr>
<td>elections.wav</td>
<td>1.711</td>
<td>4.204</td>
<td>3.061</td>
<td>3.498</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>2.569</td>
<td>2.503</td>
<td>2.288</td>
<td>1.584</td>
</tr>
<tr>
<td>letters3&amp;4.wav</td>
<td>4.276</td>
<td>4.059</td>
<td>3.438</td>
<td>3.539</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>2.935</td>
<td>0.648</td>
<td>0.670</td>
<td>0.000</td>
</tr>
<tr>
<td>vollchl.wav</td>
<td>3.215</td>
<td>3.204</td>
<td>3.111</td>
<td>3.771</td>
</tr>
</tbody>
</table>

In Figure 19, the average error magnitude of the Global Threshold Method text-based estimations is plotted as a function of the sum of the standard deviation of durations of the end-of-sentence pauses and the Total Pause Duration Error attributable to assigning pauses to the wrong group. There is a very high correlation between the timecode estimation error and averaging and pause selection errors. Additional analysis showed that the effect of choosing incorrect pauses was greater than the effect of variance of the end-of-sentence pauses. Even though the average duration of the end-of-sentence pauses is significantly greater than the average duration of the within-sentence pauses, any deviation from the ideal case where there is no overlap in the durations of the two
groups of pauses negatively affects the estimation algorithms. When the pause characteristics of a recording are not ideal, the assumption that all timecodes occur at the longest pauses, and using average values for pauses turn out to be weaknesses in the algorithm that cannot be overcome by optimizing other parameters in the algorithm. These results indicate that in order to improve timecode estimations, the process of assigning pauses to the end-of-sentences group needs to be refined and using average values for pause lengths needs to be avoided.

*Figure 19. Average Error of Global Threshold Text-based Estimation as a function of Standard Deviation of End-of-Sentence Pause Duration and the Total Pause Duration Error due to Incorrect Pause Selection.*
4.4 Local Maxima Method

The Local Maxima Method was developed to address timecode estimation errors due to averaging and incorrectly designating pauses as end-of-sentence pauses. Like the Global Threshold Method, the Local Maxima Method started with estimating caption durations using character count weighting. However, pauses are not assigned to the end-of-sentence group solely by a global duration threshold and average pause durations for end-of-sentence and within-sentence punctuation marks are not calculated. Instead, the duration of each sentence is estimated from the text-based analysis. The audio file is scanned over a limited time range centered at the end of the first sentence as estimated by the text-based analysis, thereby incorporating pause location into the selection process. The pause with maximum duration in this range is assigned to the corresponding sentence end. The estimated start of the next caption is set to the end of the chosen pause. This process continues one sentence at a time to the end of the audio file. A range of time is used to scan for pauses in order to account for errors in the CCW estimation. It is necessary to optimize the time range. If the time range is too large, there may be multiple end-of-sentence pauses to choose from. If the time range is too small, end-of-sentence pauses may not be detected. There are two parameters that determine the magnitude of the time range over which the search for pauses occurs for each caption; the minimum time range and the proportion of the preceding caption duration.

Tests were run to determine the effect of these parameters on the average timecode estimation error magnitude. Figure 20 shows the effect of minimum search range on the average timecode estimation error. The trend for both files tested is that the average error in timecode estimation is at a moderate level at very low minimum search
ranges. As the minimum search range is increased, the average error decreases to a minimum point and subsequently increases rapidly after a critical value. The optimal minimum search range observed for both files was 1.5 seconds.

![Graph showing the effect of minimum search range on average timecode error magnitude.](image)

**Figure 20.** Effect of Minimum Search Range on Average Timecode Error Magnitude.

Figure 21 shows the effect of the proportion of the estimated duration of the previous caption on average timecode estimation error. For both files tested, the average error is relatively high at low values of this parameter. As the value of this parameter increases, the average error reaches a minimum value. As the parameter increases further, the average error begins to increase rapidly. Based on these results the parameters used are 1.5 seconds and 0.25 of the estimated duration of the previous caption from text analysis.
Figure 21. Timecode Estimation Error as a function of Proportion of Estimated Duration of Previous Caption (minimum range of 1.5 s).

The average error magnitude and the improvement over the global threshold method are given in Table 9. The results indicate that except for one file, the accuracy of timecode estimation using local maxima was much better than the global threshold method. This is due in part to the Local Maxima Method allowing for variations in speech rate within a recording. When human speech rate is changed, most of the change is in pause duration (Trouvain and Grice, 1999; Zellner, 1998b). Even though the pause selection performs better on the local level than on the global level, errors in pause selection still occur in the files with overlap in the durations of within-sentence pauses and end-of-sentence pauses. Although the Local Maxima Method offers an improvement
over the Global Threshold Method, there are still issues that need to be resolved for non-ideal files.

Table 9

*Average Magnitude of Error for Local Maxima Method.*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Average Error Magnitude (s)</th>
<th>Improvement over Global Threshold (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>basraLibrary.wav</td>
<td>2.6023</td>
<td>0.3869</td>
</tr>
<tr>
<td>car.wav</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>elections.wav</td>
<td>0.1211</td>
<td>3.3768</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>0.0000</td>
<td>1.5839</td>
</tr>
<tr>
<td>letters3and4.wav</td>
<td>5.1579</td>
<td>-1.6181</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>vollch1.wav</td>
<td>1.3579</td>
<td>2.4138</td>
</tr>
</tbody>
</table>

4.5 Local Maxima Method with Automated and Manual Feedback

Figure 22 shows the timecode estimation error for basraLibrary.wav as the timeline progresses. For the first 140 seconds of the media clip, the timecode estimations are accurate. From 140 seconds to the end of the media clip, the errors are quite high. This shows that once an error occurs, the algorithm is not able to compensate and in fact, error tends to accumulate. In order to counteract error accumulation in the Local Maxima Method, two techniques (one automated, one manual) to provide feedback to the algorithm were developed and tested. The goal of using a feedback mechanism is to limit
Automated Estimation of Time Codes for Captioning 75

timecode estimation error before it accumulates rather than trying to go back and make

corrections after the algorithm has run.

Figure 22. Timecode Estimation Error for basraLibrary.wav

The automated feedback results in a lower average error for vollch1.wav but has

either no effect or a negative effect for other files. This is due to errors in the estimation

of caption durations in the text analysis and due to a faulty assumption that the text

estimated duration could provide a valid check for the pause selection. No relationships

were observed between the feedback parameters used and the correct timecodes that were

valid for all of the files used in this study. In order to provide feedback that is completely

independent of assumptions in the pause selection algorithm, a manual feedback

technique was developed and added to the Local Maxima Method. If the Local Maxima
Method accumulated three pause choices whose durations were below a global threshold, a web interface was triggered to manually determine caption timing.

The accuracy of the web interface used to manually determine timecodes was evaluated by determining the average difference between the actual beginning of a caption and the time that was input using the web interface. This was done in order to insure that manual feedback would counteract errors rather than introduce new errors into the timecode estimation process. The web interface was tested for all of the captions in all of the audio files for a total of 341 cases. The average difference for all of the media clips is 0.13 seconds. In most cases, the timecode from the manual web interface lagged behind the actual time by one to two tenths of a second because of reaction time to press the button. However, in a few cases, the manually determined time was a bit ahead of the actual time because of reacting to an audible breath intake. In all cases, the manually determined times were accurate enough so that when a closest temporal match was run, all of the measurements were accurate within 0.5 seconds.

The results of the Local Maxima Method with manual feedback are given in Table 10. There is a significant improvement in the accuracy of the algorithm for all of the files in this study. The percentage of estimations that were correct within 0.5 s was 79% for basraLibrary.wav, and ranged from 97 – 100% for the rest of the files. The higher error values for basraLibrary.wav were attributed to the higher degree of overlap in the durations of end-of-sentence pauses and other pauses in that file. The maximum error for all of the files was five seconds using the Local Maxima Method with manual feedback, compared to 15 seconds for the Local Maxima Method with no feedback and 25 seconds for the Global Threshold Method. The manual feedback rate, defined as the
number of requests for manual input divided by the total number of captions for each file,
ranged from 0.18 (8 out of 43) down to no requests for manual feedback as shown in
Table 10. A combined automated/manual approach is similar to a number of semi-
automated segmentation algorithms for audio and video. One approach was to identify
key frames with human intervention and apply automated algorithms to frames between
the key frames (Mochamad, Loy, Aoki, 2005).

Table 10

*Average Error and Improvement Provided by Manual Feedback, Manual Input Requests,
Number of Captions, Manual Feedback Rate*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Error (s)</th>
<th>Improvement (s)</th>
<th>Feedback Requests</th>
<th>Captions</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>basraLib.wav</td>
<td>0.735</td>
<td>1.867</td>
<td>8</td>
<td>43</td>
<td>0.186</td>
</tr>
<tr>
<td>car.wav</td>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td>6</td>
<td>0.000</td>
</tr>
<tr>
<td>elections.wav</td>
<td>0.106</td>
<td>0.015</td>
<td>6</td>
<td>41</td>
<td>0.146</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td>9</td>
<td>0.000</td>
</tr>
<tr>
<td>letters3&amp;4.wav</td>
<td>0.114</td>
<td>5.044</td>
<td>8</td>
<td>120</td>
<td>0.067</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td>3</td>
<td>0.000</td>
</tr>
<tr>
<td>vol1ch1.wav</td>
<td>0.085</td>
<td>1.273</td>
<td>12</td>
<td>119</td>
<td>0.101</td>
</tr>
</tbody>
</table>
4.6 Effect of Amplitude Threshold on Accuracy of Audio File Scan

When scanning the audio file, the amplitude threshold must be set properly in order to accurately distinguish between silence and speech. If the amplitude threshold is set too low, recording and background noise is misclassified as speech. If the amplitude threshold is set too high, regions of speech are misclassified as silence. The ability to automatically determine this parameter will enhance the usability of the captioning tool. In order to accomplish this goal, the audio data were further analyzed in terms of the amplitudes of various features in the recording. Methods of measuring properties of the recordings without user intervention were also investigated.

Most applications use the recording noise as a basis for determining the amplitude threshold between speech and silence in a recording (Arons, 1997). Recording noise is the low-level hiss in an audio file that is inherent in the audio recording equipment. It has been observed (Hess, 1976) that histograms of audio levels tend to have peaks corresponding to recording noise because speech signal levels change much faster with time than the relatively constant noise. Analyses were performed on the audio data in this study in order to determine if a similar relationship existed. Histograms were generated for the amplitude magnitudes of the audio data in this study by determining the maximum amplitude in every 100 ms window and sorting the maximum amplitudes.

An example of a histogram of the maximum amplitude in every 100 ms window of a recording is shown in Figure 23. The WAV audio files used in this study had 16-bit samples, so the amplitude ranged from -32768 to 32767. The histogram shows a distinct maximum at an amplitude of 80 and the associated peak continues out to an amplitude of
somewhere between 200 and 500, depending on where the transition point is defined. After the initial peak, the histogram count remains at a relatively constant low level.

For each histogram, two values were derived: the amplitude where the peak value occurred and an estimate of the amplitude corresponding to the upper onset of the peak. Finding the peak value was straightforward; it was a matter of determining the histogram bin with the highest frequency. The process to estimate the upper onset point of the initial histogram peak was to calculate the average count per histogram bin and then starting at the peak value, search for the first count less than or equal to the average count. The amplitudes from the histogram maximums, the estimated upper onset point of the histogram peak, and the recording noise measured from the WAV files for all of the audio recordings are given in Table 11.

![Maximum Amplitude Histogram for elections.wav](image)

*Figure 23. Maximum Amplitude Histogram for elections.wav.*
The histogram peak falls below the manually measured recording noise for all of the files. This is in part because the manually measured recording noise is a maximum value rather than an average. However, there is a direct relationship between the histogram peak and the recording noise as evidenced by the very high correlation between these two quantities ($R=0.9833$). The correlation between the upper onset of the peak and the recording noise is also very high ($R=0.9098$). For all of the files in this study, the upper onset of the histogram peak is greater than the manually measured recording noise. For that reason, using the upper onset of the peak is preferable to using the maximum value of the peak as an estimate of the recording noise since it provides a more conservative estimate. In order to set a threshold value for separating silence from speech, Hess (1976) found the histogram peak and derived a value above that peak. Hess

Table II

*Peak Amplitude from Histograms and Manually Measured Recording Noise from Audio Files (full scale = 32767).*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Histogram Peak</th>
<th>Upper Onset of Peak</th>
<th>Recording Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>basralib.wav</td>
<td>120</td>
<td>660</td>
<td>225</td>
</tr>
<tr>
<td>car.wav</td>
<td>140</td>
<td>220</td>
<td>154</td>
</tr>
<tr>
<td>elections.wav</td>
<td>80</td>
<td>400</td>
<td>229</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>140</td>
<td>320</td>
<td>240</td>
</tr>
<tr>
<td>letters3&amp;4.wav</td>
<td>820</td>
<td>1600</td>
<td>1146</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>160</td>
<td>280</td>
<td>246</td>
</tr>
<tr>
<td>vollchl.wav</td>
<td>780</td>
<td>2140</td>
<td>896</td>
</tr>
</tbody>
</table>
was able to use this approach because the quality of the audio signals in that study was very good and there were no background noises to account for.

Background noise is defined as non-speech signals such as rustling papers, throat clearing or breath intake. These sorts of non-speech signals were present in the audio files used in this study. The manually measured background noise and the RMS for each audio file used in this study are given in Table 12. One important observation is that the amplitude of the background noise is significantly greater than the estimates of the recording noise from the histograms. Another important observation is that the RMS is greater than the background noise level in all of the files.

Table 12

<table>
<thead>
<tr>
<th>Filename</th>
<th>Background Noise</th>
<th>RMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>basraLib.wav</td>
<td>711</td>
<td>1298</td>
</tr>
<tr>
<td>car.wav</td>
<td>275</td>
<td>1722</td>
</tr>
<tr>
<td>elections.wav</td>
<td>573</td>
<td>850</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>442</td>
<td>1595</td>
</tr>
<tr>
<td>letters3&amp;4.wav</td>
<td>2949</td>
<td>7133</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>573</td>
<td>1980</td>
</tr>
<tr>
<td>vollch1.wav</td>
<td>4915</td>
<td>6475</td>
</tr>
</tbody>
</table>
Timecode estimation was performed for each audio file using the Local Maxima Method with Manual Feedback Algorithm. The algorithm was run using the upper onset of the histogram peak and the RMS of each file in order to assess the effectiveness of each measurement as an amplitude threshold. The results of these tests are given in Table 13. The average error in timecode estimation for all of the files was lower when using the RMS as the amplitude threshold. For all but one of the files, the manual feedback request rate was lower using the RMS. In terms of both efficiency and accuracy, RMS provides better results.

Table 13

*Average Timecode Estimation Error in Local Maxima Method with Manual Feedback Using Upper Onset of Histogram Peaks and RMS as Amplitude Threshold.*

<table>
<thead>
<tr>
<th>Filename</th>
<th>Upper Onset of Peak</th>
<th>RMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (s) Requests</td>
<td>Error (s) Requests</td>
</tr>
<tr>
<td>basraLib.wav</td>
<td>0.791 8</td>
<td>0.735 8</td>
</tr>
<tr>
<td>car.wav</td>
<td>0.349 1</td>
<td>0.000 0</td>
</tr>
<tr>
<td>elections.wav</td>
<td>0.199 3</td>
<td>0.086 6</td>
</tr>
<tr>
<td>elevator.wav</td>
<td>0.055 0</td>
<td>0.000 0</td>
</tr>
<tr>
<td>letters3&amp;4.wav</td>
<td>0.370 11</td>
<td>0.114 8</td>
</tr>
<tr>
<td>spacetime.wav</td>
<td>0.269 0</td>
<td>0.000 0</td>
</tr>
<tr>
<td>vollch1.wav</td>
<td>0.155 25</td>
<td>0.085 12</td>
</tr>
</tbody>
</table>
In order to better understand the effect of amplitude threshold on timecode estimation, a series of estimations were run using amplitude thresholds from below the recording noise to near the maximum amplitude in a recording. Figure 24 is an example of the effect of amplitude threshold on timecode estimation error for the audio files used in this study. For each recording, there is a range of amplitude thresholds that result in a minimal error magnitude. If the amplitude threshold is either higher or lower, the average error begins to increase. The graph also shows the calculated RMS and the manually measured recording noise and background noise. When the amplitude threshold is lower than the recording noise, the average error magnitude increases sharply because the pause duration measurements are no longer accurate. However, there is still significant error in the timecode estimations when the amplitude threshold is at or just above the recording noise amplitude. It is not until the amplitude threshold is greater than the background noise that the error magnitude is minimized. The instances of breath intake are not a significant source of the error because they occur right before a speech signal. However, paper rustling and throat clearing contribute more to the timecode estimation error because these can occur in the middle of long silences and lead to the algorithm incorrectly splitting a long pause into two short pauses. The RMS of the audio file is in the range of minimal error. This observation holds true for all of the files used in this study.
One approach to filtering out background noise is to set the amplitude threshold close to the recording noise, and then defining another time threshold to ignore transient, short duration signals (Gan & Donaldson, 1988; Horii, 1983). This is used in Voice Over IP and speech recognition applications because it is necessary in these cases to have a very precise measure of when speech signals begin. If the RMS were used as the amplitude threshold in those applications, the beginnings and ends of speech segments would be clipped off. However, because the timing of captions only needs to be within 0.5 seconds of the audio event, and because the typical transient from above the RMS to the pause amplitude is less than 0.1 seconds in the files used in this study, the RMS of each audio file performs adequately as an amplitude threshold. Using a time threshold for ignoring signals requires choosing an arbitrary value that may or may not be valid for 

Figure 24. Effect of Amplitude Threshold on Timecode Estimation Error, car.wav
Automated Estimation of Time Codes for Captioning 85

given recording. Using the RMS to estimate the amplitude threshold for each file has the advantages of being simple and being based on direct measurements of each recording. Also, the RMS provides a more robust or safe estimate of the amplitude threshold than the actual background noise because it consistently falls in the middle of the range of minimal error.

In order to automatically set the amplitude threshold it will be necessary to perform at least two scans of the audio data. The first pass would be to calculate the RMS to set the amplitude threshold. The second pass would use the amplitude threshold for detecting pauses or silences in the recording. One issue with performing multiple reads of the audio data is that it is computationally and time intensive in PHP. Using an executable C program instead of PHP to scan the audio could alleviate this. The C program can be triggered by using the PHP exec() function and the data from a C program can be obtained and written to a database by first writing the data to a text file and subsequently using a PHP script to read the text file and input the data into a database.
Summary and Conclusions

5.1 Summary

Efforts have been made in this study to develop a process to automate estimating the timing of captions for web-based audio and video files using plain-text transcripts and their corresponding audio recordings. There were some preliminary tasks to be optimized such as parsing text transcripts into sentences, automatically determining an amplitude threshold between silence and speech in an audio recording, determining the location and duration of pauses in an audio recording, and analyzing trends between pause duration and corresponding text features. Once the preliminary analyses were accomplished, algorithms for estimating the duration of each sentence were developed. The algorithms began with text-based analyses to make an initial estimate of the duration of each sentence. Two different methods of incorporating audio data to refine those duration estimates and to match ends of sentences to specific pauses in the audio recording were developed and tested. Finally, feedback mechanisms for minimizing the error in the estimation algorithms were developed and tested.

The major findings of this study include that for parsing text transcripts into sentences, simply scanning text for end-of-sentence punctuation is not sufficient because these punctuation marks are not used just to denote sentence ends. Automated sentence boundary detection in texts is not a trivial task because periods and exclamation points serve multiple functions that can only be discerned when considering context. However, if a line break or some other formatting is used to distinguish end-of-sentence
punctuation marks, then sentence boundary detection and caption character counting is easily automated.

A second major finding of this study is that the RMS of each audio file performs adequately as an amplitude threshold for estimating caption timecodes. Using the RMS to estimate the amplitude threshold for each file has the advantages of being simple and being based on direct measurements of each recording. Analyzing the audio data and calculating the RMS of the amplitude successfully automated the step of determining an amplitude threshold between silence and speech.

Audio data analysis showed that for each audio recording taken as a whole, the durations of pauses at the ends of sentences are significantly longer than pauses within sentences and there is overlap in the durations of the two groups. However, within localized regions of recordings, the relationship between pause duration and syntactic function appears to be stronger than over the entire file. When the audio recordings were split into segments, the difference in duration between the two groups of pauses remained significant and the degree of overlap between the two groups decreased. These results indicate that in terms of successfully distinguishing between within-sentence pauses and end-of-sentence pauses on the basis of duration, it is advantageous to analyze data over localized portions of a file rather than over the entire audio file.

These observations were borne out in the results of the estimation algorithms. In the Global Threshold Method, there is a very high correlation between the timecode estimation error and averaging and pause selection errors. These errors were minimized in the Local Maxima Method. When matching pauses to end-of-sentence punctuation according to pause duration, using local individual pause duration data to estimate
caption timecodes is more accurate than using global averaged pause duration data due to variations in speech rate over the course of an audio file. However, a global pause duration threshold seems to be a useful criterion to trigger manual feedback requests.

In terms of the actual algorithms for estimating caption timecodes, it was necessary to incorporate manual feedback in a semi-automated process so that the timecode determination approached acceptable levels of error. However, the highest request rate for manual feedback was less than one out of every five captions and some files required no manual feedback at all. Manual feedback worked much better than automated feedback. This is due to errors in the estimation of caption durations in the text analysis and due to a faulty assumption that the text estimated duration could provide a valid check for the pause selection. No relationships were observed between the feedback parameters used and the correct timecodes that were valid for all of the files used in this study. The Manual Feedback is completely independent of the assumptions used in the pause selection algorithm.

5.2 Conclusions

The purpose of this study was to develop a process to automate the determination of time codes for captions from a full-text transcript and an audio track. The best procedure developed in this study is to split text transcripts into sentences using the standard captioning format of inserting a line-break after punctuation to denote ends of sentences, make an initial estimate of the duration of each sentence using Character Count Weighting (CCW), set an Amplitude Threshold between speech and silence by calculating the RMS of the amplitude of the recording, scan the audio and compile the locations and durations of pauses, use the Local Maxima Method to match ends of
Automated Estimation of Time Codes for Captioning

captions to pauses in the audio, and trigger requests for manual feedback when the results of the Local Maxima Method are ambiguous. There are two criteria that can be used to measure the success of this study, accuracy and extent of automation. For the files tested in this study, the algorithm accurately estimated the timing of 96% of the captions within 0.5 seconds. Based on the criterion of accuracy, this study was successful. The procedure has also achieved a high level of automation. The tasks of initially estimating the duration of sentences from the number of characters, determining an amplitude threshold between silence and speech, and determining the location and duration of pauses in the audio recordings is fully automated. The task of matching ends of sentences to pauses in the audio recording is semi-automated. If a recording is ideal in the sense that all end-of-sentence pauses are greater in duration than pauses within sentences, no manual feedback is required. Manual feedback is triggered in order to deal with recordings in which the durations of end-of-sentence pauses overlap with the durations of within pauses. Finally, generating web-ready formats to synchronize media and captions from the timecode estimations is fully automated.

The procedure for the automation of timecode determination developed in this study will be most advantageous in situations where a script is created before recording the audio track. In that situation, a small amount of reformatting would be necessary to prepare the text for the captioning process. Also, the algorithms developed in this study work best in those conditions because speech is more consistently structured when a speaker is reading a script or a passage from a book. The current algorithm to automatically estimate timecodes, combined with minimal manual feedback, will be useful for captioning media that contains scripted or formal speech.
5.3 Recommendations for Future Work

Future research should address the incorporation of additional time-domain audio data parameters such as zero-crossing rate and its correlation to certain phonemes. It may also be useful to investigate incorporating frequency-domain analysis of the audio data to identify phonemes. If certain classes of phonemes can be identified and correlated to the text, particularly in the vicinity of pauses, it could enhance the accuracy of automatically identifying the ends of sentences in the audio and reduce the need for manual feedback. Additional development should enhance the usability of the web interface and solve the problem of cross-browser compatibility. Also, if the algorithms and audio data scanning become more computationally intensive, the coding will need to be moved from PHP to a more time-efficient programming language such as C.
References


Automated Estimation of Time Codes for Captioning


Automated Estimation of Time Codes for Captioning 97


Automated Estimation of Time Codes for Captioning 98


Appendix A: PHP Script to Parse a Text Transcript into Sentences

```php
<?php include('dbconn.php'); ?>
<?php $projectID = $_GET['projectID']; ?>
<?php $resultParameters = mysql_query("SELECT * FROM projects WHERE id = $projectID;", $db2); ?>
<?php $rowParameters = mysql_fetch_array($resultParameters, MYSQL_ASSOC);
$textfile = $rowParameters["text"];
?>

// begin text analysis

$filesize = filesize($textfile);
$fr2 = fopen($textfile, "r");
$text = file($textfile);
$lines = count($text);
$lines = $lines - 1;
$transcript = ";
for ($j=0;$j<=$lines;$j++) {
$transcript = $transcript.$text[$j];
}

$transcript = str_replace("--", ",", $transcript);
$transcript = str_replace("", ",", $transcript);
$transcript = str_replace("._", ",", $transcript);
$wordcount = str_word_count($transcript);

$charcount = strlen($transcript);
$charcount = $charcount - 1;

$transarr = str_split($transcript);
$punccount = 0;
$puncendcount = 0;
for ($aaa=0;$aaa<=$charcount;$aaa++) {
if($transarr[$aaa] == "." OR $transarr[$aaa] == "!" OR $transarr[$aaa] == "?"{)
$puncendcount = $puncendcount + 1;
}
Automated Estimation of Time Codes for Captioning

```php
if ($transarr[$aaa] == "," OR $transarr[$aaa] == " - " OR $transarr[$aaa] == ";" OR $transarr[$aaa] == ":") {
    $punccount = $punccount + 1;
}

} //end for $aaa

$puncendcount = $puncendcount - 1; //remove last punctuation point from count

$bb = 0;
$charcount2 = 0;
for ($aa=0;$aa<=$charcount;$aa++) {
        $word[$bb] = $word[$bb].$transarr[$aa];
        $word[$bb] = trim($word[$bb]);
        $wordlength[$bb] = strlen($word[$bb]);
        $punccountarr[$bb] = substr_count($word[$bb], ",");
        $puncendcountarr[$bb] = substr_count($word[$bb], ");
        $wordcountarr[$bb] = str_word_count($word[$bb]);
        $interlexicalarr[$bb] = $wordcountarr[$bb] - 1 - $punccountarr[$bb];
        $bb = $bb + 1;
    } else {
        $word[$bb] = $word[$bb].$transarr[$aa];
    } // end if . and ; test
} // end of aa loop
fclose($fr2);
?>
```
Appendix B: PHP Script to read WAV Audio and Determine Pauses

```php
<?php
include('dbconn.php');
$projectID = $_GET['projectID'];
$result2 = mysql_query("SELECT *
FROM projects
WHERE id = $projectID;", $db2);

$row2 = mysql_fetch_array($result2, MYSQL_ASSOC);
$mediafile = $row2['media'];
$audiofile = $row2['audio'];
$textfile = $row2['text'];
$amplitudethreshold = $row2['ampThresh'];
$timethresholdms = $row2['timeThresh'];
$timethreshold = $timethresholdms/1000;
?>
<?php set_time_limit(2000);
//read audio file formatting parameters
$fr = fopen($audiofile, "rb");

$chunkid = fread($fr, 4);

$chunksizeb1 = fread($fr, 1);
$chunksizeb1 = bin2hex($chunksizeb1);
$chunksizeb2 = fread($fr, 1);
$chunksizeb2 = bin2hex($chunksizeb2);
$chunksizeb3 = fread($fr, 1);
$chunksizeb3 = bin2hex($chunksizeb3);
$chunksizeb4 = fread($fr, 1);
$chunksizeb4 = bin2hex($chunksizeb4);
$chunksize = "0x".$chunksizeb4.$chunksizeb3.$chunksizeb2.$chunksizeb1;
$chunksize = hexdec("$chunksize");

$format = fread($fr, 4);

$subchunk1id = fread($fr, 4);
$subchunk1sizeb1 = fread($fr, 1);
$subchunk1sizeb1 = bin2hex($subchunk1sizeb1);
$subchunk1sizeb2 = fread($fr, 1);
$subchunk1sizeb2 = bin2hex($subchunk1sizeb2);
$subchunk1sizeb3 = fread($fr, 1);
$subchunk1sizeb3 = bin2hex($subchunk1sizeb3);
$subchunk1sizeb4 = fread($fr, 1);
$subchunk1sizeb4 = bin2hex($subchunk1sizeb4);
$subchunk1size = "0x".$subchunk1sizeb4.$subchunk1sizeb3.$subchunk1sizeb2.$subchunk1sizeb1;
$subchunk1size = hexdec("$subchunk1size");
```
Automated Estimation of Time Codes for Captioning 102

```
$audioformatb1 = fread($fr, 1);
$audioformatb1 = bin2hex($audioformatb1);
$audioformatb2 = fread($fr, 1);
$audioformatb2 = bin2hex($audioformatb2);
$audioformat = "0x".$audioformatb2.$audioformatb1;
$audioformat = hexdec("$audioformat");

$numchannelsb1 = fread($fr, 1);
$numchannelsb1 = bin2hex($numchannelsb1);
$numchannelsb2 = fread($fr, 1);
$numchannelsb2 = bin2hex($numchannelsb2);
$numchannels = "0x".$numchannelsb2.$numchannelsb1;
$numchannels = hexdec("$numchannels");

$samplerateb1 = fread($fr, 1);
$samplerateb1 = bin2hex($samplerateb1);
$samplerateb2 = fread($fr, 1);
$samplerateb2 = bin2hex($samplerateb2);
$samplerateb3 = fread($fr, 1);
$samplerateb3 = bin2hex($samplerateb3);
$samplerateb4 = fread($fr, 1);
$samplerateb4 = bin2hex($samplerateb4);
$samplerate = "0x".$samplerateb4.$samplerateb3.$samplerateb2.$samplerateb1;
$samplerate = hexdec("$samplerate");

$byterateb1 = fread($fr, 1);
$byterateb1 = bin2hex($byterateb1);
$byterateb2 = fread($fr, 1);
$byterateb2 = bin2hex($byterateb2);
$byterateb3 = fread($fr, 1);
$byterateb3 = bin2hex($byterateb3);
$byterateb4 = fread($fr, 1);
$byterateb4 = bin2hex($byterateb4);
$byterate = "0x".$byterateb4.$byterateb3.$byterateb2.$byterateb1;
$byterate = hexdec("$byterate");

$blockalignb1 = fread($fr, 1);
$blockalignb1 = bin2hex($blockalignb1);
$blockalignb2 = fread($fr, 1);
$blockalignb2 = bin2hex($blockalignb2);
$blockalign = "0x".$blockalignb2.$blockalignb1;
$blockalign = hexdec("$blockalign");

<bitsperb1 = fread($fr, 1);
<bitsperb1 = bin2hex($bitsperb1);```
$bitsperb2 = fread($fr, 1);
$bitsperb2 = bin2hex($bitsperb2);
$bitsper = "0x".$bitsperb2.$bitsperb1;
$bitsper = hexdec("$bitsper");

$subchunk2id = fread($fr, 4);
$subchunk2sizeb1 = fread($fr, 1);
$subchunk2sizeb1 = bin2hex($subchunk2sizeb1);
$subchunk2sizeb2 = fread($fr, 1);
$subchunk2sizeb2 = bin2hex($subchunk2sizeb2);
$subchunk2sizeb3 = fread($fr, 1);
$subchunk2sizeb3 = bin2hex($subchunk2sizeb3);
$subchunk2sizeb4 = fread($fr, 1);
$subchunk2sizeb4 = bin2hex($subchunk2sizeb4);
$subchunk2size = "0x".$subchunk2sizeb4.$subchunk2sizeb3.$subchunk2sizeb2.$subchunk2sizeb1;
$subchunk2size = hexdec("$subchunk2size");

$numsamples = $subchunk2size * 8 / $bitsper / $numchannels;

// start reading audio file amplitude data

$eep = $timethreshold * 1000;
$cliptime = $numsamples/$samplerate;
$filesize = $chunksize + 8;
$samplesize = $bitsper/8;

$present = 10;
$starttime = 0;
$pausecount = 0;
$sum = 0;

for ($i=0;$i<=$numsamples; $i++){
    $time = $i/ $samplerate;
    $preceding = $present;
    $presentb1 = fread($fr, 1);
    $presentb1 = bin2hex($presentb1);
    $presentb2 = fread($fr, 1);
    $presentb2 = bin2hex($presentb2);
    $present = "0x".$presentb2.$presentb1;
    $present = hexdec("$present");

    if ($present > 32767){
        $present = $present - 65536;
    }
}
//running calculation of RMS
$sum = $sum + $present*$present;

if (abs($present) > $amplitudethreshold) {
    if (abs($preceding) <= $amplitudethreshold) {
        $difference = $endtime - $starttime;
        if ($difference > $timethreshold) {
            $lala = $difference*1000.0;
            if ($starttime >= 0) {
                // send values of detected pauses to database
                $result1 = mysql_query("INSERT INTO pauses (startTime, endTime, duration, projectID) VALUES($starttime, $endtime, $lala, $projectID);", $db2);
            }
        }
    }
}

if (abs($preceding) > $amplitudethreshold) {
    $starttime = $starttime;
}

if (abs($present) <= $amplitudethreshold) {
    if (abs($preceding) <= $amplitudethreshold) {
        $endtime = $time;
    }
    if (abs($preceding) > $amplitudethreshold) {
        $starttime = $time;
    }
}

} // end for $i

// close audio file
fclose($fr);

mysql_query("UPDATE projects SET cliptime = $cliptime WHERE id = $projectID;", $db2);
?>
Appendix C: PHP script to generate histogram and calculate RMS of audio data.

```php
<?php
include('dbconn.php');
$projectID = $_GET['projectID'];
$result2 = mysql_query("SELECT * FROM projects WHERE id = $projectID;", $db2);
$row2 = mysql_fetch_array($result2, MYSQL_ASSOC);
$mediafile = $row2['media'];
$audiofile = $row2['audio'];
$textfile = $row2['text'];
$amplitudethreshold = $row2['ampThresh'];
$timethresholdms = $row2['timeThresh'];
$timethreshold = $timethresholdms/1000;
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Read Audio Files, generate histogram and calculate RMS of audio data</title>
<link rel="stylesheet" type="text/css" href="style.css">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<?php set_time_limit(10000); //read audio file formatting parameters
$fr = fopen($audiofile, "rb");

$chunkid = fread($fr, 4);

$chunksize1 = fread($fr, 4);
$chunksize1 = bin2hex($chunksize1);
$chunksize2 = fread($fr, 4);
$chunksize2 = bin2hex($chunksize2);
$chunksize3 = fread($fr, 4);
$chunksize3 = bin2hex($chunksize3);
$chunksize4 = fread($fr, 4);
$chunksize4 = bin2hex($chunksize4);
$chunksize = "0x".bin2hex($chunksize4).bin2hex($chunksize3).bin2hex($chunksize2).bin2hex($chunksize1);
$chunksize = hexdec("$chunksize");

$format = fread($fr, 4);

$subchunk1id = fread($fr, 4);
$subchunk1size1 = fread($fr, 4);
$subchunk1size1 = bin2hex($subchunk1size1);
$subchunk1size2 = fread($fr, 4);
```
Automated Estimation of Time Codes for Captioning 106

$\text{subchunk1sizeb2} = \text{bin2hex($\text{subchunk1sizeb2}$)};
$\text{subchunk1sizeb3} = \text{fread($fr, 1$)};
$\text{subchunk1sizeb3} = \text{bin2hex($\text{subchunk1sizeb3}$)};
$\text{subchunk1sizeb4} = \text{fread($fr, 1$)};
$\text{subchunk1sizeb4} = \text{bin2hex($\text{subchunk1sizeb4}$)};
$\text{subchunk1size} = \text{"0x".$\text{subchunk1sizeb4}$.\text{subchunk1sizeb3}$.\text{subchunk1sizeb2}$.\text{subchunk1sizeb1}};
$\text{subchunk1size} = \text{hexdec("$\text{subchunk1size}$")};

$\text{audioformatb1} = \text{fread($fr, 1$)};
$\text{audioformatb1} = \text{bin2hex($\text{audioformatb1}$)};
$\text{audioformatb2} = \text{fread($fr, 1$)};
$\text{audioformatb2} = \text{bin2hex($\text{audioformatb2}$)};
$\text{audioformat} = \text{"0x".$\text{audioformatb2}$.\text{audioformatb1}};
$\text{audioformat} = \text{hexdec("$\text{audioformat}$")};

$\text{numchannelsb1} = \text{fread($fr, 1$)};
$\text{numchannelsb1} = \text{bin2hex($\text{numchannelsb1}$)};
$\text{numchannelsb2} = \text{fread($fr, 1$)};
$\text{numchannelsb2} = \text{bin2hex($\text{numchannelsb2}$)};
$\text{numchannels} = \text{"0x".$\text{numchannelsb2}$.\text{numchannelsb1}};
$\text{numchannels} = \text{hexdec("$\text{numchannels}$")};

$\text{samplerateb1} = \text{fread($fr, 1$)};
$\text{samplerateb1} = \text{bin2hex($\text{samplerateb1}$)};
$\text{samplerateb2} = \text{fread($fr, 1$)};
$\text{samplerateb2} = \text{bin2hex($\text{samplerateb2}$)};
$\text{samplerateb3} = \text{fread($fr, 1$)};
$\text{samplerateb3} = \text{bin2hex($\text{samplerateb3}$)};
$\text{samplerateb4} = \text{fread($fr, 1$)};
$\text{samplerateb4} = \text{bin2hex($\text{samplerateb4}$)};
$\text{samplerate} = \text{"0x".$\text{samplerateb4}$.\text{samplerateb3}$.\text{samplerateb2}$.\text{samplerateb1}};
$\text{samplerate} = \text{hexdec("$\text{samplerate}$")};

$\text{byterateb1} = \text{fread($fr, 1$)};
$\text{byterateb1} = \text{bin2hex($\text{byterateb1}$)};
$\text{byterateb2} = \text{fread($fr, 1$)};
$\text{byterateb2} = \text{bin2hex($\text{byterateb2}$)};
$\text{byterateb3} = \text{fread($fr, 1$)};
$\text{byterateb3} = \text{bin2hex($\text{byterateb3}$)};
$\text{byterateb4} = \text{fread($fr, 1$)};
$\text{byterateb4} = \text{bin2hex($\text{byterateb4}$)};
$\text{byterate} = \text{"0x".$\text{byterateb4}$.\text{byterateb3}$.\text{byterateb2}$.\text{byterateb1}};
$\text{byterate} = \text{hexdec("$\text{byterate}$")};

$\text{blockalignb1} = \text{fread($fr, 1$)};
Automated Estimation of Time Codes for Captioning

$blockalignb1 = bin2hex($blockalignb1);
$blockalignb2 = fread($fr, 1);
$blockalignb2 = bin2hex($blockalignb2);
$blockalign = "0x".$blockalignb2.$blockalignb1;
$blockalign = hexdec("$blockalign");

$bitsperb1 = fread($fr, 1);
$bitsperb1 = bin2hex($bitsperb1);
$bitsperb2 = fread($fr, 1);
$bitsperb2 = bin2hex($bitsperb2);
$bitsper = "0x".$bitsperb2.$bitsperb1;
$bitsper = hexdec("$bitsper");

$subchunk2id = fread($fr, 4);
$subchunk2sizeb1 = fread($fr, 1);
$subchunk2sizeb1 = bin2hex($subchunk2sizeb1);
$subchunk2sizeb2 = fread($fr, 1);
$subchunk2sizeb2 = bin2hex($subchunk2sizeb2);
$subchunk2sizeb3 = fread($fr, 1);
$subchunk2sizeb3 = bin2hex($subchunk2sizeb3);
$subchunk2sizeb4 = fread($fr, 1);
$subchunk2sizeb4 = bin2hex($subchunk2sizeb4);
$subchunk2size = "0x".$subchunk2sizeb4.$subchunk2sizeb3.$subchunk2sizeb2.$subchunk2sizeb1;
$subchunk2size = hexdec("$subchunk2size");

$numsamples = $subchunk2size * 8 / $bitsper / $numchannels;

// start reading audio file amplitude data
$eep = $timethreshold * 1000;
$cliptime = $numsamples/$samplerate;
$filesize = $chunksize + 8;
$samplesize = $bitsper/8;
$present = 10;
$starttime = 0;
$pausecount = 0;
$sum = 0;

for($bin=0;$bin<=32750;$bin=$bin+50){
$countAll[$bin] = 0;
$countMax[$bin] = 0;
}

for ($i=0;$i<=$numsamples; $i++)
	$time = $i/$samplerate;
	$preceding = $present;
Automated Estimation of Time Codes for Captioning

```php
$presentb1 = fread($fr, 1);
$presentb1 = bin2hex($presentb1);
$presentb2 = fread($fr, 1);
$presentb2 = bin2hex($presentb2);
$present = "0x".$presentb2.$presentb1;
$present = hexdec("$present");

if ($present > 32767) {
    $present = $present - 65536;
}

$sum = $sum + $present*$present;

$sdph1 = abs($present);
$sdph = $sdph1/50;
$sdph = ceil($sdph);
$sdph = $sdph * 50;
$countAll[$sdph] = $countAll[$sdph] + 1;

if($i > 0) {
    $windowCount = $i % ($samplerate/10);
    $value[$windowCount] = $sdph1;

    if($i % ($samplerate/10) == 0 OR $i == $numsamples) {
        $maxdph1 = max($value);
        $maxdph = $maxdph1/50;
        $maxdph = ceil($maxdph);
        $maxdph = $maxdph * 50;
        $countMax[$maxdph] = $countMax[$maxdph] + 1;
        $maxTot = $maxTot + 1;
    }
}
fclose($fr);

$rms = sqrt($sum/$numsamples);
echo("RMS = $rms<br>"e);
$total = 0; for($ii=0;$ii<=32760;$ii+=$ii+50) {
    $bincount = $countMax[$ii];
```
Automated Estimation of Time Codes for Captioning

```php
save($ii, $bincount); $total = $total + $bincount;
$data4 = $ii . "", "$" . $bincount . "\n"; fputs($fr6, $data4);
fclose($fr6);
echo("<br><a href='windowMax.csv'>windowMax.csv</a>";
echo("<br>Histogram total = $total");
</td></tr></table>
</body>
</html>
```
Appendix D: SQL Scripts to create Database Tables

CREATE TABLE projects ( 
  id int(11) NOT NULL AUTO_INCREMENT,
  projectName varchar(200) NOT NULL,
  media varchar(50) NOT NULL,
  audio varchar(50) NOT NULL,
  text varchar(50) NOT NULL,
  timeThresh int(11) NOT NULL,
  ampThresh int(11) NOT NULL,
  cliptime float default NULL,
  PRIMARY KEY ('id')
);

CREATE TABLE pauses ( 
  pauseID int(11) NOT NULL AUTO_INCREMENT,
  startTime float NOT NULL,
  endTime float NOT NULL,
  duration float NOT NULL,
  projectID int(11) NOT NULL,
  PRIMARY KEY (‘pauseID’)
);

CREATE TABLE timecodes ( 
  ID int(11) NOT NULL,
  manual float NOT NULL default '0',
  caption text,
  localMax float NOT NULL default '0',
  closestMatchToManual float NOT NULL default '0',
  projectID int(11) NOT NULL,
  estimation2 float NOT NULL default '0',
  global float default NULL
);
Appendix E: Sample PHP Script to Generate Timecode Data Files

```php
<?php

include("dbconn.php"); //connect to database

$projectID = $_GET['projectID'];

if ($projectID == "") {
    $projectID = $_POST['project'];
}

if($projectID <> ""){
    $result2 = mysql_query("SELECT *
FROM projects
WHERE id = $projectID;", $db2);
    $row2 = mysql_fetch_array($result2, MYSQL_ASSOC);
    $mediaFile = $row2["media"];  
    $audioFile = $row2["audio"];  
    $textFile = $row2["text"];  
    $amplitudeThreshold = $row2["ampThresh"];  
    $timeThresholdMs = $row2["timeThresh"];  

    $fr1 = fopen('timecodes.csv', 'w');

    $result3 = mysql_query("SELECT *
FROM manual
WHERE projectID = $projectID
ORDER BY ID;", $db2);

    //insert data headers
    $data = "manual,closestMatch,estimation1,estimation2\n";
    fputs($fr1, $data);

    //insert data row by row
    while ($row3 = mysql_fetch_array($result3, MYSQL_ASSOC)) {
        $actual = $row3["timecode"];  
        $closestMatch = $row3["closestMatch"];  
        $estimation = $row3["estimation"];  
        $estimation2 = $row3["estimation2"];  
        $data = "$actual,$closestMatch,$estimation,$estimation2\n";
        fputs($fr1, $data);
    } // end while $row3

    fclose($fr1);

    echo("<br><a href='timecodes.csv'>timecodes.csv</a>\n")
} //end if $projectID <> ""
Appendix F: Code to Embed Windows Player in a web page.

```php
<?php $mediafile = $_GET['mediafile']; ?>
<html>
<head>
<title>Untitled Document</title>
<link href="slider/css/winclassic.css" rel="stylesheet" type="text/css">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
body { font: MessageBox; font: Message-Box; }
input { width: 50px; text-align: middle; padding: 1px; margin-right: 15px; }
input, select, button { vertical-align: middle; }
#slider-1 { margin: 10px; width: 300px; }
#slider-2 { margin: 10px; }
#color-picker { border: 0; }
input solid rgb(90,97,90); width: 350px; height: 80px; }
</style>
<script type="text/javascript" src="slider/js/range.js"></script>
<script type="text/javascript" src="slider/js/timer.js"></script>
<script type="text/javascript" src="slider/js/slider.js"></script>

<SCRIPT language="javascript">
function startup() {
Player.controls.play(); s.setValue((a/b)*100);
}

function stop() {
Player.controls.stop();
s.setValue(0);
}

function pause() {
Player.controls.pause();
s.setValue((a/b)*100);
}

function delay() {
setTimeout("mtime()");
}</SCRIPT>

<!-- .style1 {font-family: Arial, Helvetica, sans-serif} -->
</head>
<body onLoad="delay();">
```
<table cellpadding="8">
<tr><td>
<p>
<OBJECT ID="Player" WIDTH=300 HEIGHT=300 CLASSID="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6">
<PARAM NAME="URL" VALUE="<?php echo ($mediafile); ?>">
<PARAM NAME="UIMode" VALUE="NONE">
</OBJECT>
</p></td></tr></table>
<p>
<div class="slider" id="slide-1" tabIndex="1">
<input class="slider-input" id="slide-input-1" />
</div>
<p>
<input id="h-value" type="hidden" onChange="s.setValue(parseInt(this.value))"/>
<input id="h-min" type="hidden" onChange="s.setMinimum(parseInt(this.value))"/>
<input id="h-max" type="hidden" onChange="s.setMaximum(parseInt(this.value))"/>
</p>
<script type="text/javascript">
function delay2() {
setTimeout("mtime()");
lala = 0;
}
function mtime() {
if (lala > 0) {
a = Player.controls.currentPosition;
b = Player.currentMedia.duration;
document.getElementById("h-value").value = (a/b)*100;
}
</script>
<SCRIPT>
lala = 0;
function mtime() {
if (lala > 0) {
a = Player.controls.currentPosition;
b = Player.currentMedia.duration;
document.getElementById("h-value").value = (a/b)*100;
}
</SCRIPT>
document.getElementById("h-max").value = 100;
c = (a/b)*100;
}
lala = lala + 1;
setTimeout("mtime()", 1000);
</SCRIPT>
<table cellpadding="8" width="300">
<tr><td align="center">
<FORM "form1">
<INPUT TYPE="BUTTON" NAME="BTNSTART" VALUE=""
>"OnClick="startup()">
<INPUT TYPE="BUTTON" NAME="BTNPAUSE" VALUE=" || "
OnClick="pause()">
<INPUT TYPE="BUTTON" NAME="BTNSTOP" VALUE="STOP"
OnClick="stop()">
</FORM>
</td></tr>
</table>
Automated Estimation of Time Codes for Captioning 115

Appendix G: Data Capture Page for Manually Determined Timecodes

```php
<?php
$projectID = $_GET['projectID'];
if ($projectID == "") {
    $projectID = $_POST['projectID'];
}

$thisTime = $_POST['mediaTime'];
$thisID = $_POST['id'];
$hisAnchor = trim($thisID);
include('dbconn.php');

$result1 = mysql_query("UPDATE manual SET timecode = $thisTime WHERE ID = $thisID AND projectID = $projectID;", $db2);
?>

<link href="http://www.eiu.edu/eiu.css" rel="stylesheet" type="text/css">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head> <title>Untitled Document</title> 
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<SCRIPT language="javascript">
function delayO { setTimeout( "mtimeO", 100);
}

</SCRIPT>

<style type="text/css">
<!-- .stylel {font-family: Arial, Helvetica, sans-serif} -->
</style>
</head>
<body onLoad="delay();window.location.href='#!/?php echo ($thisAnchor-1) ?>'">

<form action="dataCapture3.php" method="post" name="form2" target="dataCapture">
<table>
<tr>
<td colspan="4">
<br><span class="stylel">Clip Time </span>
<input type="text" NAME="mediaTime" value="0">
</td>
</tr>
</table>
```
Automated Estimation of Time Codes for Captioning

```php
<?php
$result0 = mysql_query("SELECT * FROM projects WHERE id = $projectID;", $db2);
$row0 = mysql_fetch_array($result0, MYSQL_ASSOC);
$projectName = $row0["projectName"];
?

<table border="1">
<tr><td colspan="7">$projectName</td></tr>
<tr bgcolor="#CCCCCC" align="center">
<td>Set<br>Timecode</td>
<td>Manual<br>timecode (s)</td>
<td>Best<br>Match (s)</td>
<td>Caption</td>
<td>Pause<br>Duration (ms)</td>
</tr>
<?php
$result1 = mysql_query("SELECT * FROM manual WHERE projectID = $projectID
ORDER BY ID;", $db2);
while ($row = mysql_fetch_array($result1, MYSQL_ASSOC)) {
$id = $row["ID"];
$timecode = $row["timecode"];
$caption = $row["caption"];
$closestMatch = $row["closestMatch"];
$result2 = mysql_query("SELECT duration FROM pauses WHERE projectID =
$projectID AND endTime LIKE $closestMatch;", $db2);
$row2 = mysql_fetch_array($result2, MYSQL_ASSOC);
$duration = $row2["duration"];
?
<?php echo ("<tr><td align="center"><a name="$id"></a><input name='id'
type='submit' value=' $id'></td><td align="center">$timecode</td><td align="center">$closestMatch</td><td>$caption</td><td>$duration</td></tr>"); ?>
<?php } ?>

<tr>
<td colspan="4"><input name="projectID" type="hidden" value="$projectID"></td></tr>
</table>
</form>
```
Perform Closest Pause Match

Back to Select Project
Appendix H: PHP Code to generate SMIL for Real Player, SMIL for QuickTime, and SAMI for Windows Media Player

```php
<?php
include("dbconn.php");
$projectID = $_GET['projectID'];
$resultParameters = mysql_query("SELECT * FROM projects WHERE id = $projectID;", $db2);
$rowParameters = mysql_fetch_array($resultParameters, MYSQL_ASSOC);
$mediafile = $rowParameters["media"];
$audiofile = $rowParameters["audio"];
$textfile = $rowParameters["text"];
$cliptime = $rowParameters["cliptime"];
$author = $rowParameters["user1"];
$projectName = $rowParameters["projectName"];
?>

<?php
$SMILcode = "<smil><head><meta name="title" content="SMIL Wrapper"/>
";
$SMILcode = $SMILcode . "<layout><root-layout background-color="black" 
height="300" width="355"/>
";
$SMILcode = $SMILcode . "<region id="videoregion" background-color="black" 
top="5" left="5" height="240" width="350"/>
";
$SMILcode = $SMILcode . "<region id="textregion" background-color="#000000" 
top="240" left="5" height="60" width="350"/>
";
$SMILcode = $SMILcode . "</layout></head><body><par>
";
$SMILcode = $SMILcode . "</par><n</video src=""$mediafile"
region=""videoregion""/>
";
$SMILcode = $SMILcode . "</n<switch><textstream
src=""$textfile.rt" region=""textregion" system-language="en" system-captions="on" 
title="english captions!" alt="english captions!"/>
"</switch>
";
$SMILcode = $SMILcode . "</par><n</body>
";
$SMILfile = "project" . $projectID . ".smil.txt"
);

$fr = fopen($SMILfile, 'w');
fputs($fr, $SMILcode);
fclose($fr);
?>

<?php
$query = mysql_query("SELECT * FROM manual WHERE projectID = $projectID ORDER BY ID;", $db2);
```
Automated Estimation of Time Codes for Captioning

$rtCode = "<window bgcolor="#000000" wordwrap="true"
duration="$cliptime"><font size="4" face="Arial"
color="#FFFFFF"><center>

$qtTextCode = "\{QTtext\} {font: Arial} {justify: center} {size: 12} {backcolor:0, 0, 0} {timescale: $cliptime} {width: 320} {height: 60}\\n"

$SAMlcode = "<SAMI><HEAD><TITLE>SAMI Captions</TITLE><STYLE TYPE="text/css">" ;
$SAMlcode = $SAMlcode . "!-- P {font-size: 1.2ems;font-family: Arial;font-weight: normal;color: #FFFFFF;background-color: #000000;text-align: center;};" ;
$SAMlcode = $SAMlcode . $.ENUSCC { name: English; lang: EN-US-CC; } -- >";</STYLE></HEAD><BODY>"

$lasttimecode = 0;
while($row = mysql_fetch_array($query, MYSQL_ASSOC))

$timecode = $row['estimation'];
$qtwhole = floor($timecode);
$qtfrac = $timecode - $qtwhole;
$qtframes = $qtfrac*0.3;
$qtframes = round($qtframes, 2);
$timecodeqt = $qtwhole + $qtframes;
if($timecodeqt < 10) {
$timecodeqt = "0" . $timecodeqt;
}
$timecodeSAMI = round($timecode*1000);
$timecodenonprop = $timecode - $lasttimecode;
$caption = $row['caption'];

$rtCode = $rtCode . "<time begin=$timecode/>\n$caption\n";
$qtTextCode = $qtTextCode . "[00:00:$timecodeqt]\n$caption\n";
$SAMlcode = $SAMlcode . "<SYNC Start=$timecodeSAMI><P Class=ENUSCC><br>$caption</P></SYNC>";

$lasttimecode = $timecode;

$rtCode = $rtCode . "</center> </font> </window>";
$qtTextCode = $qtTextCode . "[00:00:$cliptime]\n";
$SAMlcode = $SAMlcode . "</BODY></SAMI>";
Automated Estimation of Time Codes for Captioning

```
<?php
qt:autoplay="true" qt:time-slider="true">

<head>

<title>

<mediafile>

<region id="videoregion" background-color="black" top="0" left="0" height="240" width="350"/>

<region id="textregion" background-color="#000000" top="240" left="0" height="66" width="350"/>

</layout>

</head>

<body>

<par>

</body>

</smil>";
```

Automated Estimation of Time Codes for Captioning

```php
$SMILqtCode = $SMILqtCode . "<!-- CAPTIONS -->
<textstream src="$textfile.qt.txt" region="textregion" system-language="en" system-captions="on" title="english captions" alt="english captions"/>
";

$SMILqtfile = "project" . $projectID . ".qt.smil.txt";

$fr3 = fopen($SMILqtfile, 'w');
fputs($fr3, $SMILqtCode);
fclose($fr3);

?>

<?php
$ASXcode = "<asx version="3.0">
<abstract>This is the shows abstract</abstract>
<title>$projectName</title>
<author>$author</author>
<copyright>(c) 2002 - company name</copyright>
<entry>

<ref href="$mediafile?SAMI=$SAMlfile2">
<abstract>This is the clips abstract</abstract>
<title>$projectName</title>
<author>$author</author>

<copyright>( c) 2000 - company name</copyright>
</entry>
</asx> ";

$ASXfile = "project" . $projectID . ".asx";

$fr6 = fopen($ASXfile, 'w');
fputs($fr6, $ASXcode);
fclose($fr6);

?>

<!--The following HTML with embedded PHP sets up links to download and preview the captioned media generated above -->

<table bgcolor="#eeeeee"<p class="style 1 ">
Right click on the links below and <strong>Save</strong> to an appropriate location. <br> The downloaded files need to be in the <strong>SAME</strong> folder as the media file.</p>

<p class="style 1 ">
<a href="?php echo ($SMILfile); ?" target="_blank">$SMILfile</a>
</p>

<p class="style 1 ">
<a href="?php echo ($rtFile); ?">$rtFile</a>
</p>
```
<?php echo ($rtFile); ?>

SMIL for QuickTime

<?php echo ($SAMIfile); ?>

Windows Media: SAMI

ASX file: (left click to Preview in Internet Explorer)