Web Site for Collaboration and Task Distribution in Video Caption Creation - C3

Dept. of CIS - Senior Design 2009-2010

Evan Benshetler
ebensh@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Jonathan M. Smith
jms@cis.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

ABSTRACT

The purpose of this project is to expedite the process of subtitle file creation through two means: distribution of labor and intelligent language analysis. At present, subtitle file creation is inefficient, as there is little to no collaborative subtitling aid software in existence. This forces one user to sit and navigate through an entire video to caption it. In addition, the task of finding each individual caption’s beginning and end point in the video’s time sequence is laborious and time consuming.

We propose that the range of time markers for each caption’s beginning and end can be narrowed down significantly through acoustical analysis of an audio track with the aid of a user generated transcript. Also, by distributing the task of captioning each segment of video among many users of a collaborative website, the time required by each individual is drastically reduced. Such a system would expedite the subtitle creation process significantly, reduce the time requirement for each individual user, and therefore lead to many more captioned videos being publicly available, greatly benefiting those who use them.

1. INTRODUCTION

Approximately 6 million of 7.5 million people in the UK who use subtitles are not hearing impaired in any way, which demonstrates the popularity and usefulness of subtitles even for those who do not need them [14]. Subtitles aid in the viewer’s understanding, as well as providing access to media that would otherwise be unattainable for people with hearing impairments. Unfortunately, there are a few problems with providing captions for all programming. To understand the barrier caption providers face at present, one must first understand precisely what a caption is.

Captions are defined as “The title of a scene, the text of a speech, etc., superimposed on the film and projected onto the screen” by Merriam-Webster, which further defines subtitles as “the text of dialogue, speeches, operas, etc., translated into another language and projected on the lower part of the screen”, in this case a translation from spoken dialogue to printed [11]. There are two main types of captions: open and closed. Open captions are permanently displayed over the video stream, whereas closed captions can be shown or hidden at the end user’s discretion. Our project will be dealing primarily with closed captions in the form of a subtitle file separate from a video file. One example of a common closed subtitle format is the SRT format, which looks like:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:02:17.440 --&gt; 00:02:20,375</td>
</tr>
<tr>
<td>2</td>
<td>Senator, we’re making our final approach into Coruscant.</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>00:02:20,476 --&gt; 00:02:22,501</td>
</tr>
<tr>
<td>8</td>
<td>Very good, Lieutenant.</td>
</tr>
</tbody>
</table>

Regardless of its specific format a subtitle file typically contains many separate captions in chronological order, where each caption is a textual string accompanied by a start time and end time. YouTube, for example, uses the SRT format minus the first row, eliminating the need for the sequence number and instead relying on the time codes in the second line. When a media player displaying a video with its accompanying subtitles reaches a caption’s start time, it displays the caption; when it reaches the caption’s end time, it removes it from the screen. Please note that all of this is entirely different from the “line 21” captioning that occurs with modern television programs. “Line 21” captioning is “transmitted on line 21 of field 1 of the vertical blanking interval of television signals”, which only “TV broadcast receivers with pictures screens larger than 33 cm (13 in) or larger” are required to support and display. The focus of C3 is creating subtitle files for computer video files, which is completely removed from the problem of television program captioning [3].

While there are many software tools at present that aid a person in creating subtitles for a video, there are still two main problems:

1. It is a very time intensive process.
2. Some techniques require human speech recognition - either of the video dialog or of a person re-voicing the dialog - for better speech recognition accuracy.
Regardless of what software is used, the key steps of subtitle creation are:

1. Identifying start and end time of caption.
2. Writing caption text.
3. Placing caption visually (x, y) coordinates (only in certain formats).
4. Repeat.

There are many modern subtitle creation aids that expedite this process. One that we have chosen to highlight later on is Subtitle Horse [16]. Some ways in which these tools help are:

1. Hotkeys to allow faster start/end time marking.
2. Output to many common subtitle formats, allowing one subtitle creation job to be reused for different media.
3. More intuitive graphical interface to allow easier subtitle editing (order, start/end time, auto numbering, etc.).

We have used tools similar to this one in the past. Our personal experiences have lead us to believe that the time consumption of the subtitling process is heaviest in steps 1 and 2 above: identifying caption start and end times, and writing out the caption text. This method would address these topics and attempt to reduce the overall time required to perform them, which would lead to the creation of more subtitle files and better caption availability.

2. RELATED WORK - CAPTIONING

There have been many previous proposals of more efficient caption creation methods. Since there are so many different situations requiring captions, there is, of course, no one “correct” method. Even in as narrow a field as using pure speech recognition to create captions, there are multiple methods. In an article by BBC’s R&D employee Dr. Evans, the following techniques that BBC uses for speech recognition captioning (depending on whether a recording and a transcript are available at the time of subtitle creation) are listed [7]:

<table>
<thead>
<tr>
<th>Recording Available?</th>
<th>Transcript Available?</th>
<th>Type of Speech Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>speaker-independent</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>speaker-specific followed by speaker-independent</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>speaker-specific</td>
</tr>
</tbody>
</table>

For example, a live broadcast would not have a recording nor (most likely) a transcript available, so speaker-specific speech recognition is a necessity. A speech recognition algorithm trained for the voice of a baritone male from Texas will not work nearly as well for a soprano female from Norway. Given the variety of real-life situations that are broadcast, it is no surprise how many different methods there are for even this narrow subset.

2.1 Subtitle-Horse.org

There are many caption creation aiding programs available on the Internet. One of these was Subtitle Horse, a program that focuses more on straightforward subtitle creation and not at all on voice recognition [16]. It is web-based and meant to be used by one caption creator at a time. Subtitle Horse plays an uploaded video back to the caption creator, allowing them to pause, seek through the video with fine-grained control, write captions quickly, and move forward rapidly to the next possible caption. After we experimented with it for a short while, we found the hotkeys were extremely useful for rapid subtitle creation and the export functionality allowed for various formats of output. Though extremely basic in functionality, it is an excellent program were one user looking to create a simple subtitle file for a short video.

Subtitle Horse is still lacking key functionality that we feel our program would supplement well. While one user running Subtitle Horse might create 100 captions for a single video, we cut down the number of captions per user required by distributing that work out among many users, reducing the amount of work time needed by each individual. Instead of working to determine start and end points in the video, which Subtitle Horse only provides hotkeys for without automatic discovery or even narrowing down, we feel that some intelligent automation can be accomplished. By analyzing a video’s accompanying audio stream for silences, for example, we could possibly cut out large chunks of silence, reducing the amount of audio/video stream the captioning user must browse. Programs like Subtitle Horse certainly still have their place, however. By using a program such as Subtitle Horse to edit the final version of captions produced by our system, we are presented with a polished tool for editing said captions without reinventing the wheel.

2.2 BBC

An article published by BBC’s R&D employee Dr. Evans discusses interesting steps taken by BBC to caption different television broadcasts [7]. We have already shown their break down of speech recognition methods, but there is more detail to be discussed. The central method used is speech recognition, but the variety of ways in which it is employed is impressive. By examining whether or not a recording is available and whether or not a transcript is available, BBC can then decide among three different forms of speech recognition: speaker-independent, speaker-specific followed by speaker-independent, and speaker-specific only. In situations where a recording and a transcript are available, such as offline captioning and in-house for a broadcast station, the method that BBC uses is called Assisted Subtitling (AS). This method is convenient for subtitled purposes because it is simpler to line up the timing of the audio with the accompanying transcript, instead of doing full speech to text recognition. This is different from our proposed method due to the fact that we do not have a script available to us, forcing our program and users to recreate the transcript of the video in subtitle form.

When the captioning is being done offline, but there is no transcript available (a situation similar to our own) BBC then employs speaker-specific followed by speaker-independent methods. They call the former “Script Capture”, which uses a voice recognition engine from IBM that is able to track multiple voices, such as in a television episode, with reason-
ably accurate recognition. An operator manually marks up the video with scene and character identification tags. This helps the speech recognition engine identify which characters are on the screen at what times, enabling higher accuracy rates. The tags also provide more information to aid temporal caption alignment in the Assisted Subtitling process once the transcript is recreated, now with helpful character and scene annotations. Though C3 does not reach into manual script mark up as BBC does, it is interesting to consider as a possible future expansion upon our project.

The final method BBC employs is speaker-specific voice recognition. This method is used for live broadcasts with no pre-recording and no transcript available. An employee revoices the dialogue into a speech-to-text recognition engine that has been trained to their voice in real time, eliminating variations in voice and background noise, and creating a much more accurate speech recognition result.

Unfortunately, this last method does not scale well, but BBC has done some interesting work here with distribution of work effort. Employees sitting at so-called “KLive” workstations connected by the Internet can swap turns, giving each other a break and allowing them to work collaboratively and remotely. Our project follows a similar collaboration paradigm, but distributes the work among many users who are not necessarily live, nor using speech recognition on their end.

2.3 Real-Time Closed-Captioning Using Speech Recognition

Another research article by Toru Imai et al. discusses revoicing dialogue for speech recognition, much the same as BBC [5]. One important observation the authors make is that “Large-vocabulary continuous speech recognition can now be found in several applications, though it does not work as well as human perception and its target domain in each application is still limited”. We agree whole heartedly, which is why we try to distribute the work of speech recognition to our user base. Most people fluent in English will have no difficulty listening to and transcribing an audio track into written English, since there is no translation to be done. As human beings, we have no trouble filtering out speech from background noise in most cases, making us wonderful speech recognition engines.

2.4 Efficient method for producing off-line closed captions

Further complex and comprehensive solutions exist, such as the patent for an “Efficient method for producing off-line closed captions”. Of all the patents, academic papers, and software we encountered on the subject, this was the most similar to our proposed system.

Van Thong et al. put forth a process that starts by classifying the audio to remove dialogue-lacking spans of times [6]. By transcribing the spoken dialogue and using “time of event keystrokes” Van Thong et al.’s system is able to match up dialogue slices with time slices. At this point, the system aligns the text into the slice, and then formats the final result into a completed closed caption.

The system which Van Thong suggests uses the caption creator’s key stroke timing to approximate the times of the spoken dialog. This introduces a completely different set of problems, such as if the caption creator slows down, different caption creators for different videos, and so on. It again provides a good rough estimate, but unless there is an exact correlation between typing and spoken words, it is really their later alignment algorithms that are doing the core of the work.

We believe that we can make this system better by ignoring the rate and times at which users input their subtitles, and using an absolute comparison based on silence and speech recognition to align a transcription with the video. This allows us to distribute our work among many users, a key focus of our project. Further, by doing statistical analysis of completed slice subtitles and overlapping slices intelligently, we believe that we can narrow down subtitles start and end times to within a second of sentences beginning and ending. Without having a user manually adding precise time tags, we feel that our method will return the most accurate timing results possible.

3. RECENT NEWS IN THE CC FIELD

Since the publishing of our initial C3 Proposal Paper, there have been two major updates that must be discussed. The first is legislation pertaining to Internet video captioning, and the second is a possible solution in a very close vein to C3.

3.1 H.R. 6320

H.R. 6320, more commonly referred to as the “Twenty-first Century Communications and Video Accessibility Act of 2008”, was first introduced to the House of Representatives in 2008. Though omitted from our initial proposal this is not due to lack of relevance, but rather an unfortunate oversight as it is still in the process of the House and has not yet been ratified. There is an extremely important portion of this bill that pertains directly to the C3 system: “(A) Within 18 months after the date of date of enactment of the Twenty-first Century Communications and Video Accessibility Act of 2008, the Commission shall prescribe regulations that include an appropriate schedule of deadlines for the provision of closed captioning of video programming distributed to the public over the Internet” [4]. This essentially states that once H.R. 6320 is passed, the FCC will have 18 months to create regulations for closed captioning of Internet-based video. It is excellent news for users of closed captions across the world and displays a definitive need for new captioning systems, as the near future might require that all videos publicly accessible be equally accessible with captioning. The C3 system is not the only one attempting to race the bill to ratification however; Google’s YouTube is taking extremely large strides in the correct direction.

3.2 Automatic captions in YouTube

One of the original concepts for C3 was that it be captioned using purely speech recognition, after which users could collaboratively edit the caption files (which would surely not be perfect). It was rapidly discarded after realizing that most videos have sound tracks, background noise, and other factors that severely hamper speech recognition. Instead of fully automatic speech recognition, C3 hands this work off to human beings, who are excellent speech recognizers.

Google’s YouTube recently added a fully automatic captioning feature called “Auto-caps” however, following the principle of “even when they’re off, they can still helpful” [8]. The automatic captioning feature uses Google Voice’s speech to text engine, which was originally used to transcribe voice
mails into text. At present there are 12 partner “channels” on YouTube that will be beta testing the feature, and their sample videos are impressive. Google also released a second feature, known as “auto-timing”, which will take a full transcript without timing information and attempt to align the transcript to the video at the appropriate times. This is essentially a mirror of the “post” processing stage of C3, in which each slice was to be time aligned in case slice captions needed to be divided into multiple segments, for example. It was a future feature, and though we were a bit surprised to hear of Google doing the same work as us it only serves to show that this service is definitely needed. A system such as C3 would not work with the volume of videos that YouTube handles (over 23 hours of new video every minute), but we feel that our system will lead to much higher accuracy for when “off [but] still helpful” just isn’t enough.

4. RELATED WORK - CROWDSOURCING

A term that the reader may or may not be familiar with is “crowdsourcing”, a buzzword that refers to using human beings to perform tasks that computers are lousy at [9]. Though computers are not necessarily lousy at speech recognition, there is a lot left to be desired in this field. As an example, computer speech recognition cannot identify terms outside of its lexicon, which decreases accuracy from already less-than-desired values. It should be plain that an integral part of C3’s nature is crowdsourcing. By offloading the work of speech to text processing onto website users with idle cycles we can drastically increase accuracy of subtitle text, especially in videos with background noise or interference that regular speech to text processors have trouble with. Two excellent examples of the success that crowdsourcing can achieve are reCAPTCHA and Foldit.

4.1 reCAPTCHA

The original CAPTCHA was used to test whether a user was a spambot or not by having the user do visual Optical Character Recognition (OCR). Its successor reCAPTCHA keeps this goal, but achieves a second one simultaneously - that of crowdsourcing OCR on words that traditional computer OCR failed on [17]. The original CAPTCHA displayed one word that had been visually scrambled to attempt to deceived spam bots and ward them off. The user would type the word as they saw it and the CAPTCHA system would verify them. Of course these words are often heavily garbled and so the user must be given multiple attempts, but spam bots have a great deal of trouble with the CAPTCHA word images.

Its crowdsourcing sequel reCAPTCHA added a new wrinkle to the authentication system by requiring the user to perform visual OCR on two words instead of one. The first word in the image was a test exactly like the original CAPTCHA to determine whether or not the user was a spam bot, the second word though actually pulled the user response into a database, and uses the user’s input as the caption! To ensure that the user is not a spam bot that got lucky on the first word or a user who correctly captioned the first word but missed on the second there is a certain amount of redundancy involved. Even with this redundancy the reCAPTCHA system had fully OCR’d over 20 years of The New York Times archives in 2009, and they are still working at it. The reCAPTCHA system is a perfect example of crowdsourcing succeeding at an activity that computers had already been proven to be lousy at, or at least bringing a 90%+ OCR accuracy up to a full 100%.

4.2 Foldit

Rosetta@home [13] and Foldit [12] are both examples of distributed work achieving a common goal: simulating protein fold configurations. Rosetta@home is an example of distributed computing, in a sense an older crowdsourcing, one that uses many computers to perform embarrassingly parallel computations. Foldit is more relevant to C3, as it directly uses crowdsourcing. Whereas reCAPTCHA uses crowdsourcing under the guise of proving a user is human, Foldit instead takes the angle of creating a game as a frontier. In this game users exercise their brains during their life’s idle cycles to attempt to find optimal folding configurations of proteins by folding them.

All of the data sourced from the game and its users is aggregated by Foldit’s creators and later analyzed for useful protein folding information. Due to the many degrees of freedom in even a simple protein’s fold computers have a terribly difficult time at performing these calculations, they are lousy at it. By crowdsourcing and using the end user’s insight the game’s authors can move their research forward much further than they would using only computers. We hope that Microsoft will consider replacing Solitaire with something that crowdsources like Foldit in the near future.

5. SYSTEM MODEL

Now that the reader is familiar with the two worlds C3 brings together, captioning and crowdsourcing, we will describe how the C3 system works. There are three main components in the C3 system, shown in Fig. 1. The “prep” phase takes in uploaded videos, then analyzes and slices them. Next is the “caption” phase, in which users are presented with a slice of video and a box in which to type a caption. Last is the “post” phase, in which all captions from a video are combined into a full subtitle file for exporting. Occasionally, the “post” phase will decide that a caption is wrong / incomplete, in which case the system loops back to the “caption” phase in an iterative process.

![Figure 1: C3 System Model](image)

There is a lot of modularity in the C3 system. It follows the Unix model of piping together many small, specific-purpose, efficient programs to get a nice end result. There were a number of reasons for creating such a flow, the primary being that the more information a user can give us about their video, the easier it is to incorporate. A few other reasons are that it’s easier to optimize individual components, much easier to debug them, and make it far more modular for the future. The Prep phase is an excellent demonstration of this is as can be seen in Fig. 2.

Each component has a single specific task it performs. The Lister program was initially going to be a “speech / no speech” tagger generously donated to our project by Ben Sapp. It has since been replaced by a plain Voice Activity Detector (VAD) written in Matlab, which means that it
now detects “silence / no silence” as opposed to specifically speech. It could just as easily be replaced by any other split identifying mechanism, such as visual scene change detection. The only common element that it needs is an output list of times to begin and end a slice, which can then be converted to our standard XML input format.

The Caption phase, shown in Fig. 3, is extremely straightforward. A slice is selected from tblSlices, currently from the set of slices which do not have any captions yet. The user watches and listens to the slice while typing its caption, then submits the caption, pushing it into our database. Selecting only uncaptioned slices facilitates the fastest subtitle file turnaround time, but removes redundancy and accuracy checking. The back end currently supports multiple captions per slice, however the frontend does not display already-captioned slices for additional captioning. In future revisions multiple captions will be submitted and compared for each slice, allowing for error checking and more accurate time sampling.

6. SYSTEM IMPLEMENTATION

6.1 Lister

Starting with the “prep” system on the back end, the first component encountered is the Lister, shown in figure Fig. 4. The Lister is currently a distribution of logical highs and lows based on signal energy. If there is a lot of energy in the recording it is taken as logical high, meaning it is not silence, and the opposite is true for low. All of this processing is done in MatLab, which lends itself to WAV (uncompressed) audio digital signal processing. It outputs a list of logical silence / no silence times, which are then run through a converter program that outputs our standard XML slice information format. The Lister’s accuracy is discussed later in section 7, System Performance.

6.2 Slice Information (SliceInfo)

In the current iteration of C3 there is no automatic functionality for uploading slice information, and so the user must rely on the system’s Lister to do slice information. If the user wrote a utility to create slice information every time they press the space bar while they are watching a video, essentially creating the slice information for us, then their information might be more accurate than a basic silence / no silence detector. Since the emphasis of the C3 project was on distributing caption work to the users and not perfectly aligning slice times, allowing the user to upload their own XML file fell to the bottom of the priority list. For others who implement similar projects though this would be a good area to look into. Our XML format, shown in Fig. 5, is public and easy to adopt.

6.3 Slicer

Next comes the Slicer, seen in Fig. 6. The Slicer is written in Python, which lends itself to easy XML file parsing. Its task is to take a given SliceInfo XML file and output two script files: a Windows .BAT batch file (shell script on Linux) and a MySQL .SQL script file. The former contains a sequence of calls to run FFmpeg with specific command line parameters that will slice the input video file up into appropriate slice video files [15]. The latter is passed into MySQL from the command line and adds information about the source video to tblVideos, as well as an entry for each slice into tblSlices.

6.4 FFmpeg Slicing

The decision to use FFmpeg stemmed from a desire to make the C3 system cross-platform compatible, robust with respect to multiple video formats, and ease of use from the command line. By invoking FFmpeg as in Fig. 7, we extract
and store each individual slice as its own video file. By recompressing the slice files, we save a significant amount of server storage space, and due to FFmpeg’s efficiency (see section 7) the processing time is certainly manageable and worth the trade off.

6.5 C3 Database
Once the slice files are created, an entry for each must be created and linked to their source video in the C3 database, the layout of which is displayed in Fig. 8. The database itself is run on MySQL, but any backend SQL server would be able to support C3’s operations [1]. The specific reasons for choosing MySQL were mainly to keep it cross-platform (unlike MSSQL), its extremely thorough and available documentation, and the ease with which it can be set up with packages such as WampServer [2].

6.6 Web Interface
The “caption” phase uses a set of PHP pages and an open source (GPL 3) Adobe Flash video streamer applet called Flowplayer [10]. Combining these technologies together, we display a selected slice from the C3 database along with an empty text area. The user enters the slice’s caption into the text area and hits submit, at which point the caption is saved into the tblCaptions table, and another slice is played for them to caption. This process continues to loop until either the database runs out of slices to caption or the user navigates away.

6.7 Monitoring and Downloading
At any point during the captioning process, a user may view and/or download the best caption file available, equivalent to proceeding to the Post phase. This prevents waiting for one last slice to be captioned and gives the user an estimate of when the captioning will be done. A convenience feature of C3 is a status page on which all videos in the system are listed along with a percentage of slices captioned for each video. The videos are separated into two categories, those that the user uploaded and those that were uploaded by other C3 users. In the current C3 iteration only one caption per slice is entered into the database. For those implementing similar systems to C3, one might consider an additional review function of watching the video and having the top 3 voted captions display along with their slice. In this way the downloading user could create their own custom caption file based on the captions that they feel are best suited to their audience.

7. SYSTEM PERFORMANCE
Before discussing the system as a whole we should discuss the performance of the individual parts. While the Lister’s performance
The C3 system has now seen many videos fully captioned, both by users associated with the project and not. The largest test performed using C3 was with five users logged in simultaneously. Ideally this number would be magnitudes greater, but the test still showed how efficient the system could be. When many users were captioning the same video the C3 system performed admirably, and the “crowdsourcing” effect was readily apparent. A video that was five minutes long took approximately two and a half minutes to fully caption with 99% accuracy, once the “Prep” phase was completed on it. Before then the C3 system had only been tested with videos one minute or so in length, and never with more than two users simultaneously. Fortunately the system passed that “litmus test” with flying colors, proving that our hard work had paid off.

As a result of the limited number of samples, statistical data on accuracy of the captions is quite sparse. Our users who captioned for us had no prior experience captioning but were in general skilled typists, and achieved 98% accuracy against a “golden standard” subtitle file - that is, the script

<?xml version='1.0' encoding='UTF-8'?><video>
  <info>
    <filename>familypicnic</filename>
    <extension>mpg</extension>
    <title>Our Family Picnic</title>
  </info>
  <slice begin='39' length='6' />
  <slice begin='48' length='8' />
  ...
</video>

Figure 5: A sample SliceInfo XML file

Figure 6: Slicer

Figure 7: Invoking FFmpeg with slicing parameters

Figure 8: C3 Database Layout
that the author was reading from in our test videos. This seems promising, and also taught us that users were able to keep up with the pace of the video and never needed to watch a slice a second time. The short length of the slices also seemed to help the user with short term memory recall, allowing the user to have a higher efficiency and less wasted time rewinding and fast forwarding the video. This of course varies user to user, and the reader should bear in mind that our users were all University of Pennsylvania students with English as their primary language.

In a controlled test on an early version of C3 the back end efficiency of the Prep process was tested over ten runs and averaged together. An example video file was a 22.6 MB MPEG, 65 second long, 640x480 resolution video. C3 cut it into 12 slices totaling 78 seconds with overlapping slice edges - each slice had one second added to its originally specified length. The total file size despite the added length was 4.38 MB combined, and the drop in quality was “not really noticeable” according to our test user.

The run time of slicing the video and inserting entries into the local MySQL server table for each slice as well as the video itself was 21.47 seconds, averaged over ten runs. This equates to close to a quarter of the length of the source video to fully process it from start to finish. Assuming a maximum YouTube video length of 10 minutes, C3 will take approximately 2.5 minutes to have their slices completely created and in our database, ready to be captioned. We feel this is certainly a reasonable length of time.

After all twelve slices were captioned, the Post process execution time was negligible. It took less than two seconds, averaged over ten runs, to run locally on the database server, a personal laptop of one of the C3 creators.

The Caption phase would benefit from further explanation. In our test the user that captioned our videos was almost always able to keep up with a real time playback of slice video, the exception being a twelve second slice. Once our user provided us with the feedback that they couldn’t keep up with such a long slice we immediately modified our Slicer to break up slices longer than nine seconds into multiple equally spaced slices.

With an average slice length of 6.5 seconds, a basic optimization of the system helped our user caption faster. Once the slice captioning PHP page was set to automatically loop back onto itself upon caption submission until the entire video was captioned (the user couldn’t gracefully quit except by navigating away), said user could spend a total of approximately 2.0 times the duration of the slice captioning it. That figure includes time refreshing and loading the page, which we feel is impressive even without further optimization. As the user became used to the refreshing and workflow he achieved sub-(2.0 × slice length) times, but the average smoothed this out to above a 2.0 ratio. We feel that with a very small amount of training, rapid typists could achieve a 1.5 ratio, meaning to caption a 6 second slice would take them 9 seconds from the page loading to the next slice being presented.

8. NOVEL FEATURES, REMAINING WORK

The C3 system has a clean pipeline to allow for future feature additions and we have implemented some of the features we thought we should include in C3 1.0. There are always more novel features to be implemented on a project such as C3 though.

One feature that we already implemented is post-processing combining and splitting of captions. What this means is that captions that have too few words within a short enough time threshold will be combined into one caption. This takes care of any Lister accidents that might have created slices that are too short or incorrectly divided. The counterpart to this is splitting up long captions, those that have too much text.

The caption is divided into multiple captions with equal time slices to make for easier viewing on the video.

Another post-processing feature we have implemented is the fixing of overlapping times of captions when providing the subtitle file to the user. In the current iteration of C3 a second is automatically added to all slices to prevent words being cut off, though this causes caption overlap when viewing the final video. In post-processing we remove that extra second, ensuring that captions do not overlap on the screen, excluding situations where the overlap is more than a second. By excluding overlaps of more than a second from this fix, we allow for more flexibility on custom ListInfo .XML files that users upload, and also allow for future redundancy features (overlapping slices to narrow down caption timing, for example).

In future revisions of C3 we would like to implement redundancy, having multiple captions per slice. This would improve the quality of captions when C3 is released into the real world where not all users are students at the University of Pennsylvania. Another feature to implement would be user moderation, having users watch the entire video instead of just slices and approving or marking down captions as their slices are played in the video. Again, crowdsourced control could provide a much higher accuracy once the system is made public.

9. RESULTS

The C3 system works, and works well. It is not a system that would be used by all of Google’s YouTube, one of C3’s original goals, due to their sheer volume of data. Consider, however, that YouTube video uploads are limited to 10 minutes for regular users [18]. Recall that in C3’s largest test a five minute video took five users with moderate C3 experience only half the length of the video, two and a half minutes, to fully caption it. Now consider that on a popular “partner” channel on YouTube there might be 300 views on a three minute video in less than ten minutes, and that number might jump to well over 800,000 views in less than two days. Community channel, a YouTube partner and video uploader, often has videos that see these view numbers.
The potential for massive crowdsourcing is there and ripe for the taking, with minor modifications to websites such as YouTube. The C3 project has proven this system to be viable, entertaining, and practical enough to consider implementing on a larger-than-academic research scale.

10. REFERENCES