Richard Gatarski
Stockholm 2010-10-04

Basic Live Captioning Application
open captions for Vidblaster, spec 2

By Richard Gatarski

Introduction

This project, led by Richard Gatarski with Westreamu AB, is partly funded by The Swedish Post and Telecom Agency (PTS). Our aim is develop a web based client for live open captioning served by Vidblaster, a Windows-based live video production tool.

About live captioning

There are basically two ways in which live produced video streams can be combined with captions (subtitles). In both cases the captions has to be entered by a subtitler/stenographer into a client application that can send the text lines to a captioning server.

Open captioning means that the captions are always visible, thus burnt into the video. This implies that the video mixing software must be able to act as the captioning server. The streaming service then simply broadcast the video mix that includes the burnt in captions.

Closed captioning means that the captions are separate from the video stream. The streaming service, or the video player, acts as the captioning server and hides or displays captions on the video.

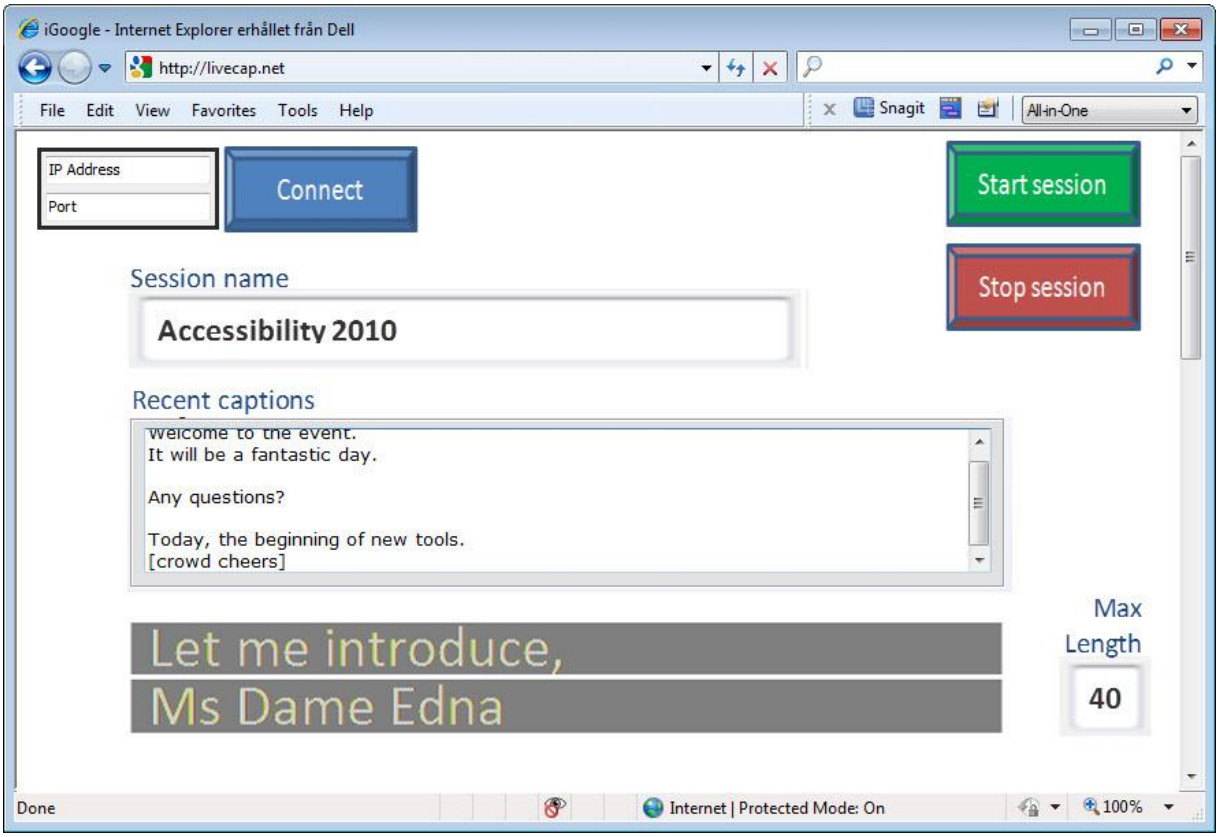
Remote captioning means that subtitler use the captioning client from any remote location. In the case of open captioning this means that they must be able to watch the event in real time.

Table of content

Richard Gatarski
Stockholm 2010-10-04

Scenario illustrating the intended use

This example shows a typical scenario where the web based captioning client (mocked up in the first illustration below) is used together with Vidblaster (screen shot in second illustration).



The video producer finds out the IP number of the captioning server (Vidblaster PC).

The subtitler opens up the captioning client in a web browser, enters session info, and via IP connects to the captioning server.

The subtitler click “Start session” and begin typing. As the typing goes on, the captions are sent by the web client to the Vidblaster server which displays them on the video (see illustration left, unfortunately wrong text).

When the session is over the subtitler clicks “Stop session”, all captions have been logged.

Current ambition

As a first step we would like to develop a fully functional application, with some limitations:

1. Support open captioning only
2. Text is sent from client to server on a word by word basis
3. Vidblaster Broadcast live video production software acts as the server
4. The subtitler is at the local event venue and use a web based client application

Later steps, to keep in mind for future developments, are likely (in no specific order):

- Predefined text support
  An option for the stenographer to specify a number of predefined texts (e.g. the name of a presenter or an organization), inserted by function keys and/or just by typing an abbreviation.
- Suggest an open protocol for live captioning
  For use by other video mixing software and streaming services.
- Web page as caption client
  Instead of sending the captions to server specific to a video mixing software, a web page acts as the client. This client takes the form of a web page with a predefined image (eg 640x480) with the text blocks over a colored background (eg green). This image can then be overlaid on the video (perhaps using chroma key) by means of a screen capture performed by the video mixing software.
- Closed captioning
  One way is to develop a plugin to JW Player that acts as the captioning server.
  Another way is to stimulate streaming providers to support our protocol.
- Remote support
  The client application can show live video from the event. For open captioning, in real time (Skype?); for closed captioning (any streaming service).

Tricky issues

For most events we cannot provide a public fixed IP to the Vidblaser computer, which implies that the client-server communication is limited to a local network (LAN). Furthermore, the client application has to be hosted on a web server outside the LAN. In this situation, will the client's web browser be able to send TCP over the LAN to the computer running Vidblaster?

Vidblaster Broadcast is rather expensive, and we cannot afford buying a license just for the development of the client application. As a solution I will have Vidblaster Broadcaster running on our machine, and provide a public IP number to the developer for testing purposes.

Richard Gatarski
Stockholm 2010-10-04

About Vidblaster's API

Vidblaster (www.vidblaster.com) is "*a state-of-the-art live video production tool that allows you to easily broadcast or record live events up to HD quality. It is an affordable multi-camera real-time software solution and it is very easy to use. […] With VidBlaster you can stream or record multiple live camera feeds together with pre-recorded video, audio and graphics. It also allows you to enhance your video production by including transitions, overlays and multiple video effects*".

From beta version 1.20 Vidblaster support a simple API as work-in-progress, see discussion thread http://forum.vidblaster.com/showthread.php?t=1121. The API is only enabled in the most expensive Broadcast version of Vidblaster (approx 1 500 Euro+VAT)

Initially the API supports a simple way to write text into the "Video overlay module". This functionality is documented in the software's help section (all versions). Basically a module in Vidblaster acts as a TCP server, connected through port 9998 on the IP-number of the computer running Vidblaster.

Currently the API supports the following commands:

- **List**
  Returns a list of the handles of all modules that are currently loaded. A handle is the module title without spaces, e.g. Camera1.

- **read** handle setting
  Reads setting from module with corresponding handle.

- **write** handle parameter [size]
  Writes data to the parameter of the module with the corresponding handle. Parameter can be text or png, when using png another argument size needs to be added indicating the size of the data to follow.

Note, use "\n" (character backslash followed by character n) in order to send text as two lines using the write command. For example:

write videooverlay1 This is the first line\nand this is the second line.

As an example Vidblaster offers a limited Windows application client called "RemoteText". It can be downloaded from http://forum.vidblaster.com/showthread.php?t=1145. The package includes all source code and a compiled .exe program, ready to run. Note that the application must be closed before it can connect to another IP address.

Requirements for the client application

The first version of the client application needs to fulfill a few basic requirements.

Open source

This project is mainly funded by PTS, The Swedish Post and Telecom Agency (www.pts.se/en-gb). They require that the first version of the application code and its documentation is provided as open source. Of course giving credit to the developers as well as the funder.

Web based

The application must be web based, not require any software installation in the client computer.

Coded for flexibility

The Vidblaster API is work in progress. Thus reprogramming the functions for interacting with it must be easy to do. For example, if a command or a response on the server is changed the places where that command is used in the code should be limited. In a similar fashion the application should be written so that it easily can be developed to support other API protocols/servers.

Furthermore, steps beyond the current ambition should be considered in the coding structure.

Session logging

The application must log all captions typed into the client during a session. This log file must be available as UTF-8 encoded in the SubRip format, see http://en.wikipedia.org/wiki/SubRip. The log should be automatic, and saved for every new caption entered in case of unexpected abortion. Here is an example of how a SubRib file looks with two caption blocks looks like. The first caption block contains one line, the second block contains two lines:

1
00:00:02,000 --> 00:00:06,200
Hello and welcome to Accessibility 2010.

2
00:00:10,800 --> 00:00:15,300
Today marks the beginning of new developments.
[crowd cheers]

Log files should be automatically named (space/national characters removed) as:
[event name]_[date]_[start time].srt, eg “Accessibility2010_2010-09-09_132312.srt”

User interface

The items below are the most important parts in the user interface. Later on, when the client application functions sufficiently, we can discuss the layout and aesthetics. The illustration above is only a mockup to indicate how the user interface may be designed.

Static input fields (to be defined by the user upon startup)

- Server IP number
- Port number (default 9998)
- Session name (40 chars)
- Text line length (default 40)

Selections

- Single-choice, one of the handles from the Overlay graphic modules in the Vidblaster server (this selection is missing in the mockup above).

Dynamic input fields

- Current captions
  Two text lines constituting current caption block (maxed to “text line length”)
  Placed on the bottom of the screen, in large readable font size.

Display window

- Recent captions
  Scrolling text, placed above “current captions”, displaying the captions most recently logged.

Buttons (the last two can be grayed, see “About saving the log files” below).

- Connect (makes connection to Vidblaster server)
- Start session
- Stop session
- (Download session , opens up a save file as dialog)
- (Remove session, opens up a list of sessions that can be deleted)

Basic flowchart for the client

This is a rudimentary flowchart for the client application when it is started and runs.

1. Make sure static fields are filled in properly
2. Wait for server connection (user press [Connect to server])
3. Poll server for available Video Overlay modules (list command)
4. Present those modules to the user, and force user select one (default none)
5. Wait for [Start session]
6. Check that a video overlay module has been selected
7. Get, send and log captions (see below)
8. Until [Stop session] is pressed, go to 7.

About getting the captions and their timings

Each “caption block” is typed in by the subtitler as one or two lines of text. For every new word (indicated by its following <space> or <enter>) the current caption block should immediately be sent to the server. Please see an example from SVT how we want it to look like: (password “opencap”) http://vimeo.com/15539401.

Word wrap the first line if it’s number of characters exceeds “text line length”. The second line might be empty, that is the subtitler just wrote one line and hit <Enter> twice. As soon as <Enter> is hit for the second line, the caption should be moved to the recent captions window.

All times should be set relative to when [Start session] was pressed, giving 00:00:00,000. This should also reset the caption block number counter to “1”. The in-time for each caption in the log is when the first space after the first word is entered. The out-time should be set to in-time plus 3 seconds if the block contains one line, or 6 seconds if two lines. If out-time his higher than next caption’s in-time, then set out-time to next caption’s in-time minus 200ms.

As soon as the out-time is determined the caption should be logged. In the case connection to the server is lost (server not responding), the client should go on recording captions. And somehow warn the user that the connection is lost.

About saving the log files

As the log files are automatically saved on the web server, we can initially manage them through FTP. In later versions of the client application we can develop functions for downloading and deleting the log files. This includes making sure that files that have not been downloaded cannot be deleted.