Lessons in Dialogue System Deployment

Anton Leuski and **Ron Artstein**
USC Institute for Creative Technologies
12015 E Waterfront Dr
Los Angeles, CA 90094, USA
{leuski|artstein}@ict.usc.edu

Abstract

We analyze deployment of an interactive dialogue system in an environment where deep technical expertise might not be readily available. The initial version was created using a collection of research tools. We summarize a number of challenges with its deployment at two museums and describe a new system that simplifies the installation and user interface; reduces reliance on 3rd-party software; and provides a robust data collection mechanism.

1 Introduction

New Dimensions in Testimonies (NDT) is a dialogue system that allows for two-way communication with a person who is not available for conversation in real time: a large set of statements is prepared in advance, and users access these statements through natural conversation that mimics face-to-face interaction (Artstein et al., 2014). Users interact with a recording of Holocaust survivor Pinchas Gutter. The system listens to their questions, selects and plays back Mr. Gutter's responses from a collection of video clips. We deployed the system at the Illinois Holocaust Museum and Education Center in Skokie since March 2015 (Traum et al., 2015a), where a museum docent relays questions from a large group audience to the system. The system was also installed for a few months at the U.S. Holocaust Memorial Museum in Washington, DC, and was demonstrated at other locations by our collaborators from the USC Shoah Foundation (SFI).

The installation proved to be a successful teaching aid: student gains were reported in interest in historical topics, critical thinking, and knowledge of issues going on in the world. The NDT system provided an engaging and emotional experience (Traum et al., 2015b). However, we discovered a number of issues with the system maintenance and support: system installation was a delicate process, there were issues maintaining the 3rd party system dependencies, the system user interface tended to overwhelm and confuse the operators, and reliable data collection proved to be a challenge.

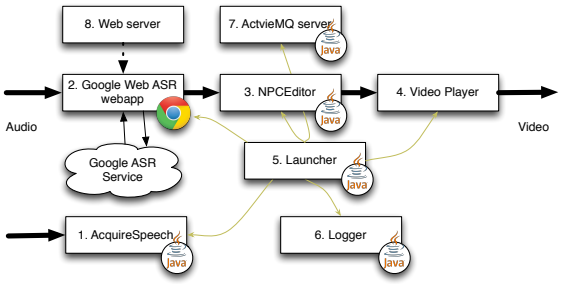

Figure 1: The initial NDT system architecture.

The lessons we learned from the deployment of the NDT system led us back to the drawing board. We created a new version of the NDT system that we call Alfred. Our goal was to make the system easier to install and maintain. We looked to simplify and streamline the user interface; create a better data collection and archiving mechanism; develop support for multiple survivor databases; and optimize the system for better performance. This paper describes the initial NDT system architecture and compares Alfred's design to it. We enumerate the challenges with encountered in the initial deployment and discuss how we addressed each challenge in Alfred.

2 Deployment Challenges

The initial NDT system was created from components of the Virtual Human Toolkit (Hartholt et al., 2013). It consists of 9 applications running on a single computer (Figure 1).

Proceedings of the SIGDIAL 2017 Conference, pages 352–355,
Saarbrücken, Germany, 15-17 August 2017.

The NDT system listens to the user's audio streaming from a microphone. Two components handle the audio stream: AcquireSpeech (1) records the user's audio to a file for later analysis, and the Google Web ASR webapp (2) sends the audio to Google speech recognition services, receives the speech transcription, and forwards the text to the language understanding module. The webapp is loaded from a web server (8) running in the background. We chose the Google speech service because it has been shown to be highly effective and robust (Morbini et al., 2013). At the time of the system development, the only way to access the Google ASR was to use the Web Speech API in the Chrome web browser[1]; we thus had to include Chrome as an additional component of the NDT system (the ninth component, not explicitly shown on Figure 1).

The language understanding and dialogue management are handled by NPCEditor (3) (Leuski and Traum, 2011). It uses a statistical classifier to analyze the speech transcript and selects the appropriate response from a collection of Mr. Gutter's video clips. It passes the clip identifier to a custom Video Player (4), which handles on-the-fly video composition such as the crossfade effect between clips and custom backgrounds for the video.

The interprocess communication between individual components is handled by messages that flow through the ActiveMQ message server (7). Logger (6) records and stores all the messages for further analysis. Finally, Launcher (5) starts and terminates the individual components. The video clips and the language data are packaged with the system components inside the Launcher application. A typical NDT installation is run on a 15-inch MacBook Pro, connected via HDMI to an external monitor or television.

Architecture The multi-component design stems from the origins of the VH Toolkit as a research and development enviromnent, where individual components can be swapped as needed. If a component crashes, the rest of the system continues to work. This design created several issues for museum deployment: the system was slow to start as each component had to be loaded and initialized separately, and if a component crashed, the system appeared to be running but stopped responding. This confused system operators, many of whom were museum volunteers with little technical training.

To simplify operation while preserving the multi-component design, Alfred appears as a single application, but internally it integrates a number of dynamically loaded plugins corresponding to individual parts of the architecture (Figure 2). The plugins deal with audio acquisition, speech recognition, response selection, video playback, logging, and external communication. The plugins use an internal messaging protocol to exchange information while isolating internal plugin details from each other. The Alfred application framework also maintains an internal database (a whiteboard) where the plugins can share data about the current state of the execution.

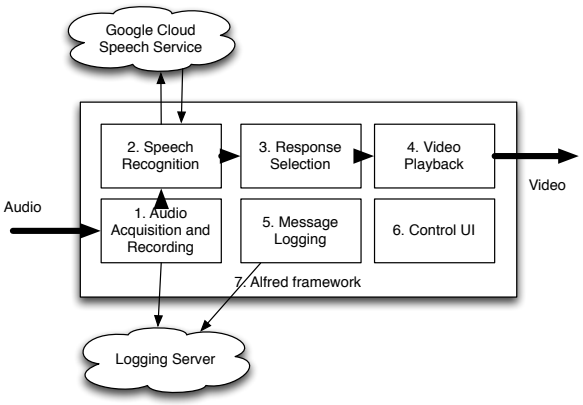

Figure 2: The Alfred NDT system architecture. Each box represents a dynamically loaded plugin that shares code in the Alfred framework.

Installation The original system installation and maintenance process is complicated by the need to install and maintain system-level 3rd party dependencies: Java SDK for Components 1, 3, 5, 6, and 7, and Google Chrome browser and Apache web server for Component 2. Configuring both web and ActiveMQ servers requires modifying the OS configuration. Google Chrome's update mechanism runs in background without user's control and can break the system without warning. This issue came to light when the NDT system stopped working one morning after the web browser updated itself overnight.

While many of the original toolkit components are cross-platform, the installation packaging for the NDT system was specific to MacOS, as both the museum staff and SFI personnel showed that they are more familiar with that environment. We therefore created Alfred as a native MacOS appli-

[1] http://tinyurl.com/mxdocae

cation. It has all the traditional UI features that a native application has. It is installed by downloading an archive file, unpacking and dragging the application icon into place. It does not require administrative level OS privileges for installation. It can be removed by dragging the application icon into the trash. Alfred has no external dependencies that we do not control – no Java runtime or Google Chrome to maintain. Alfred integrates an auto-update mechanism that checks our server at regular intervals and prompts the user to update the application if a new version is available.

User Interface Each original VH Toolkit component was created as a research tool, with its own user interface (UI). As the components were developed independently, the UIs are not consistent across them. During the system startup each component presents its own window interface. Additionally, as all the components in the system are built to be cross-platform using Java or C libraries, the UI elements in the components often do not match what a user is expecting from a native application. The number of windows and the amount of information presented in the windows overwhelm unexperienced system operators.

To simplify the UI design, we separated the user interface into regular and expert configurations. In the regular configuration, Alfred presents a single window with the survivor's video. This mode is all that is required to interact with the system. The expert mode provides an additional window with detailed information from individual plugins. The window contains tabs; each tab corresponds to a UI plugin that displays concise information summary in the tab itself, with space for more detailed information in the pane associated with the tab. For example, a UI plugin corresponding to the audio acquisition module displays the current audio power level in its tab, while providing user controls for selecting the audio source and toggling the audio recording in its pane. A number of UI plugins are provided, including one that supports a Wizard-of-Oz interface (Artstein et al., 2015).

The expert mode is disabled by default, but can be displayed on request. The expert interface is implemented as a web app that runs in a browser window, to allow observation and control from another computer. The connection is done via the standard http protocol so it is capable of crossing most firewalls without issues.

Performance The overall NDT system performance was acceptable while running on a top-of-the-line laptop. However, we encountered several challenges. Firstly, the system operators reported that the system would stop responding to the user's input for short amounts of time. We traced the issue to a Java garbage collection process effectively pausing the system at random times. Secondly, the Video Player would occasionally stutter during clip transitions. That issue was attributed to the open source, cross-platform OpenCV[2] library used for video decoding, which was not efficient enough for video playback of high resolution clips. Finally, Google Web Speech API is non-standard and poorly documented. For example, while we could request transcription in US English, that feature never worked reliably; when installing and demonstrating the system in Canada or United Kingdom, Google Chrome would detect the computer location, and the speech recognition result would default to Canadian or British spellings, throwing off the language understanding component.

As a native application, Alfred does not suffer from garbage collection issues. Some native components show higher efficiency than Java-based counterparts. For example, we re-implemented the NPCEditor text classification and dialogue management algorithms as a C++ library, and the implementation is noticeably more efficient. We continue to author the language classifiers in NPCEditor, and convert the final files into the Alfred format before deployment.

We replaced the video decoding OpenCV library with AVFoundation – the native macOS media framework. We were able to optimize the playback, decrease the computational requirements by a factor of three, and eliminate playback stutter.

Alfred uses the platform-native Google Cloud Speech API library.[3] The speech recognition plugin streams the audio from the acquisition plugin directly to the Google Cloud server. The native Google Cloud API is efficient, robust, and allows us to control the spelling variety reliably.

Scaling The initial NDT system stores both the software and the data together into a single application. Our assumption was that it would provide a single package that encapsulated the relevant pieces in one place. However, after the suc-

[2] http://opencv.org
[3] https://cloud.google.com/speech/

cess of the initial installation, the decision was made to extend the project by recording 11 additional Holocaust survivors, allowing the museum docents to switch between them. In Alfred, each survivor database is a document bundle – a folder masquerading as a single file. The document contains the video clips, the language database, and the dialogue manager scripts. Alfred opens the survivor document and loads the required information initializing the individual plugins. The document interface is a window that presents the survivor video on the screen. Closing the window closes the document and unloads its resources from memory. Switching between survivors is as easy as closing a document and opening another.

Data Collection The NDT system records both the user's audio and the inter-component messages as log files which are stored locally on the computer. The logs are used to monitor the system, evaluate its performance, and adapt the response selection algorithm. Our intention was to download the logs from the machine at the museum at regular intervals. However, access to these logs proved to be a challenge as the computer was located behind a firewall. Alfred archives both the user utterance audio and the inter-plugin messages and uploads them to our server automatically. The files are uploaded as soon as they are created. If the upload fails or the network is unavailable, Alfred attempts to upload the files again at a later time. Alfred uses a lossless codec to archive the audio, which results in files that approximately four times smaller than the files produced by the initial NDT system.

3 Conclusion

In this paper we described our analysis of an interactive dialogue system deployment in an environment where deep technical expertise might not be readily available. The initial version was created using a collection of research components and deployed at two museums. As the result of observations from the deployment, we designed a new system architecture to simplify and streamline the system installation and user interface; create a better data collection and archiving mechanism; develop support for multiple dialogue databases; and optimize the system for better performance.

We have deployed a beta version of the Alfred system both at SFI and Illinois earlier this year. The response was overwhelmingly positive: our users love the improved performance, simplified interface, and support for multiple survivor databases. We had no reports of system performance issues and the data is being collected on our servers automatically.

Acknowledgments

New Dimensions in Testimony is a collaboration between the USC Institute for Creative Technologies, the USC Shoah Foundation, and Conscience Display, and was made possible by generous donations from private foundations and individuals. We are extremely grateful to the Illinois Holocaust Museum and Education Center for hosting the exhibit and providing invaluable feedback.

References

Ron Artstein, Anton Leuski, Heather Maio, Tomer Mor-Barak, Carla Gordon, and David Traum. 2015. How many utterances are needed to support time-offset interaction? In *Proceedings of FLAIRS 28*.

Ron Artstein, David Traum, Oleg Alexander, Anton Leuski, Andrew Jones, Kallirroi Georgila, Paul Debevec, William Swartout, Heather Maio, and Stephen Smith. 2014. Time-offset interaction with a Holocaust survivor. In *Proceedings of IUI'14*. pages 163–168.

Arno Hartholt, David Traum, Stacy C. Marsella, Ari Shapiro, Giota Stratou, Anton Leuski, Louis-Philippe Morency, and Jonathan Gratch. 2013. All together now: Introducing the Virtual Human toolkit. In *Proceedings of IVA'13*. Edinburgh, UK.

Anton Leuski and David Traum. 2011. NPCEditor: Creating virtual human dialogue using information retrieval techniques. *AI Magazine* 32(2):42–56.

Fabrizio Morbini, Kartik Audhkhasi, Kenji Sagae, Ron Artstein, Doğan Can, Panayiotis Georgiou, Shri Narayanan, Anton Leuski, and David Traum. 2013. Which ASR should I choose for my dialogue system? In *Proceedings of the SIGDIAL'13*. Metz, France, pages 394–403.

David Traum, Kallirroi Georgila, Ron Artstein, and Anton Leuski. 2015a. Evaluating spoken dialogue processing for time-offset interaction. In *Proceedings of SIGDIAL'15*. Prague, Czech Republic.

David Traum, Andrew Jones, Kia Hays, Heather Maio, Oleg Alexander, Ron Artstein, Paul Debevec, Alesia Gainer, Kallirroi Georgila, Kathleen Haase, Karen Jungblut, Anton Leuski, Stephen Smith, and William Swartout. 2015b. New dimensions in testimony: Digitally preserving a holocaust survivor's interactive storytelling. In *Proceedings of ICIDS'15*. Copenhagen, Denmark, pages 269–281.