Web-Based Speech Data Collection and Annotation

Christoph Draxler

Institut für Phonetik und Sprachliche Kommunikation
Ludwig-Maximilians-Universität München
Munich, Germany
draxler@phonetik.uni-muenchen.de

Abstract

The WWW is a ubiquitous, mature communication infrastructure for business and scientific information interchange. Since 1997, the Bavarian Archive for SpeechSignals (BAS) has been developing and using web-based annotation tools for large-scale speech databases. Recently it has developed an application for recording speech via the WWW.

Both the annotation and the recording tools are now integrated into a web application for the Ph@ttSessionz speech database collection project. The goals of Ph@ttSessionz are a) to demonstrate the feasibility of WWW-based speech recording and annotation, and b) to collect a database of 1000 adolescent German speakers for the development of speech-driven applications and devices. The recordings are performed in more than 35 public schools all over Germany.

This paper will describe the recording and annotation software and discuss the technical problems that had to be overcome in the speech database collection.

1. Introduction

Since its beginning in 1990, the WWW has evolved into a ubiquitous communication network, providing the basis for many traditional and new commercial operations, personal and business information interchange, and scientific collaboration.

In many scientific areas, the WWW has become an indispensable tool for collaborative work. The decoding of the human genome, simulations of the life-cycles of stars in astronomy, or modelling the world's climate require sharing of resources, data and results, and the WWW has played an important role here.

In the area of speech databases, the WWW offers unique opportunities: for highly specialized data collections, e.g. recordings of native speakers of endangered languages or recordings that require extremely sophisticated technical setup such as X-Ray Microbeam or 3D electro-magnetic articulography, the few experts in the world can share their data independently of their geographic location. For mass data, e.g. large-scale speech databases for industrial speech technology development, processing this data can be distributed to many different sites to be performed in parallel.

Collaborative work was attractive in the late 1980s when computing resources were limited [1]. However, the increased storage capacity and processing power of modern desktop PCs has reduced the need to share resources, and as a consequence, most database creation and annotation tools in use today are standalone tools, with little or no support for distributed or collaborative work [2, 3, 4, 5].

In recent years, however, collaborative work has been given more attention again, but due to technical limitations it has focused on the annotation of speech. The BAS has been collecting both small and large speech databases since 1995. The German SpeechDat data collections of more than 6000 speakers were the starting point for the development of Web-based annotation tools [6, 7]. Since then, the BAS has continued the development of these tools, and recently other labs have presented similar solutions [8, 9].

In this paper, I will present the web based speech recording tool SpeechRecorder and the annotation framework WebTranscribe. Both tools are incorporated into a speech database web application used in the current speech data collection project Ph@ttSessionz.

2. Technology

2.1. WWW

The WWW is a combination of four basic technologies: hypertext, resource identifiers, markup languages, and the client-server architecture. The communication protocol underlying the WWW is the hypertext transport protocol *http* [10], uniform resource locators (URI) identify resources in the WWW [11], and HTML, XHTML and XML are the most wide-spread markup languages in use today [12, 13, 14].

2.2. Client-server architecture and the WWW

In a *client-server architecture*, a client requests data from a server. The server either satisfies the client request by sending a file from the local file system, a database management system, or by calling external applications to compute the requested data.

In the WWW, the client request is formulated as a URI. Parameters to the client are encoded in the URI according to the cgi-specification. The server returns either HTML pages for browser display, or data to be interpreted either by plug-ins within the browser or external viewer applications on the client.

In a *web application*, the server provides a set of services in a given application domain, and the client provides a graphical front end to these services. Just like in the WWW, client and server communicate via http, but unlike in the WWW, the client does not necessarily have to be a browser – it can be an application dedicated to the given task.

Java Web Start is a technology to deploy platform independent applications via the WWW. In Java Web Start, a client downloads the application and registers it for execution in a controlled run time environment. The application is then started just like any other local application. The only difference is that during startup it checks whether a new version is available on the WWW. If so, then this new version is downloaded and ex-

ecuted, otherwise the previous version. Thus there is no more need to explicity distribute software to users – they receive it automatically when using the application.

The major advantage of Java Web Start compared to applets is that it is completely independent of web browsers. With applets, the many browser versions, the inconsistent support for Java run time environments (JRE), and the various HTML standards for incorporating applets are a constant source of problems.

2.3. A new view on speech databases

A speech database consists of signal data, transcription and annotation data, and meta data. The transcription and annotation describe the contents of the signal data, the meta data describes the database structure, and contains information on the participants, the annotation schemes, the encoding formats, and documentation such as database specifications, validation reports, etc. [15, 16, 17, 18].

The classical view of speech databases is that of a *product*: creating a database is equivalent to building a product, using a database requires purchasing or licensing the database; it is then shipped to the customer either physically or electronically.

An alternative view, proposed here, is to regard a speech database as a *service*: a server provides the data storage, access services, and data adminstration, and clients access the server to either add to the database or extract information from it. A speech database no longer is static – it evolves over time: it may be extended by additional signal data, enhanced by additional annotation levels, etc.

The client-server architecture is particularly well suited for speech databases: the central server is responsible for the data storage, and it offers access to its data in the form of services. Clients request services from the server, and return their processing results to the server. For example, recording clients request prompt items from the server and return the recorded signal data, annotation clients request the next item to be annotated and return the annotation text, and exploitation clients request extracts of the database via search forms and display the data (fig. 1).

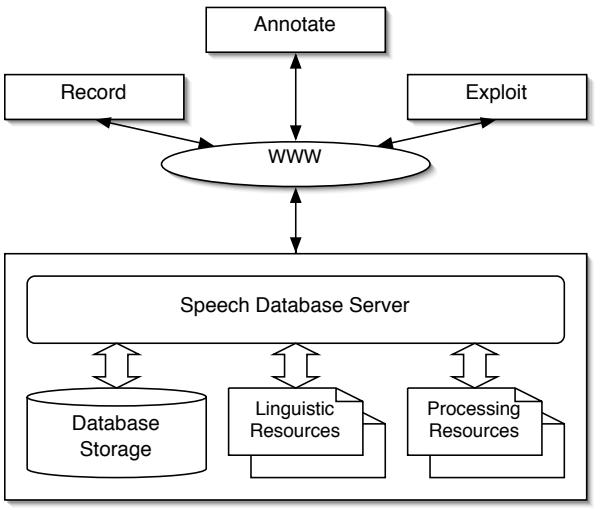

Figure 1: A client-server architecture for speech databases

The client-server architecture allows a natural distribution of resources and workload: shared resources such as signal and annotation data, or lexicons, are held centrally, while processing is distributed to the clients. New resources on the server, e.g. updates to the lexicon, new annotation levels, etc., can be made available to all clients immediately. For efficient and secure storage, database management systems can be used on the server. Furthermore, since all resources are held in one location, access policies can be enforced to control access to the data.

Note that the concept of a database being a service is not entirely new: corpus linguists have a long tradition of searching and browsing large text corpora (cf. e.g. [19, 20, 21, 23, 22], meta data has been online in the language agencies' or worldwide catalogs [24, 25, 26], and annotation systems based on the web have been described in the literature [6, 7, 9, 27]. However, to my knowledge, a consistent and unified view of speech databases in terms of web-based services has not yet been presented elsewhere.

3. SpeechRecorder

SpeechRecorder is an application for script-based speech recordings [28]. Its major features are

- sophisticated recording script and fine-grain recording protocol
- text, image and audio prompts
- multi-channel recordings local or via the WWW
- platform independence
- separate views for experimenter and speaker

In SpeechRecorder, recordings are organized in projects. A project consists of a speaker database, a number of recording sessions, and a given technical configuration.

3.1. Recording script and protocol

A recording session is divided into recording sections, each section contains a number of individual recording items. A recording section specifies the order in which the individual items are recorded (*sequential* or *random*), and how recording progress is controlled (*manual*, *autoprogress*, or *automatic*).

The recording script is defined in a simple XML document (fig. 2).

A recording is divided into four phases:

1. idle: system waiting for action command
2. prerecording: recording starts with inactive prompt
3. recording: recording with active prompt
4. postrecording: recording continues with inactive prompt

The phase *prerecording* is most commonly used to record environment noise prior to the speaker's voice, or to capture barge-in speech. *Postrecording* is used to record environment noise after the speaker has finished speaking, or to avoid truncating the signal, e.g. due to premature click on the stop button.

The phases *prerecording*, *recording*and *postrecording* are governed by timers; *recording* can be terminated by user interaction, e.g. mouse click. Each phase change is logged to a recording log file automatically.

The current phase is displayed on the screen via a traffic light (fig. 3).

```
<!ELEMENT session
   (metadata?, recordingscript)>

<!ELEMENT recordingscript
   (section)+>

<!ELEMENT section
   (nonrecording | recording)+>

<!ELEMENT nonrecording (mediaitem)>

<!ELEMENT recording
   (recinstructions?, recprompt,
    reccomment?)>

<!ELEMENT recinstructions (#PCDATA) >
<!ELEMENT recprompt (mediaitem)>
<!ELEMENT reccomment (#PCDATA)>

<!ELEMENT mediaitem (#PCDATA)*>
```

Figure 2: DTD for SpeechRecorder recording script; attributes not shown

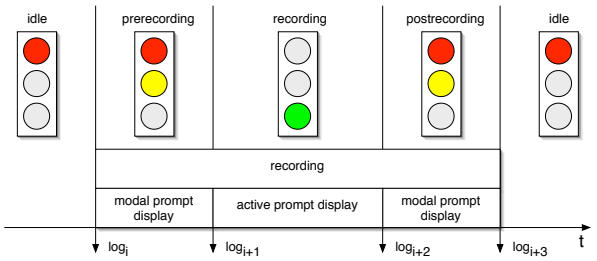

Figure 3: SpeechRecorder recording phases

3.2. Prompts

SpeechRecorder currently allows three types of prompts: plain or formatted text prompts, image prompts, and audio prompts.

Text prompts are used to elicit read speech, or to record spontaneous responses to written questions, e.g. ”how did you get here today?”. In text prompts, the standard character encodings of Java are supported, i.e. Unicode in UTF-8, UTF-16, and the ISO-8859 encoding. Formatted text, e.g. HTML or RTF documents, may be used to achieve specific layouts and to highlight parts of the text, e.g. by color or font.

Non-text prompts are used where text prompts are not desired, e.g. to elicit dialect speech, or when speakers cannot (yet) read, e.g. to record children or illiterate speakers. Images are a very powerful means of eliciting spontaneous speech.

Audio prompts may also be used to elicit free or non-scripted speech, but also for ”repeat after me” speech or to prime speakers.

Each prompt consists of an instruction to the speaker, the prompt itself, and a prompt description visible only to the experimenter. This description helps the experimenter to check whether the speaker has effectively produced the requested utterance.

A prompt item can be preceded by a beep, and the recording may be terminated via silence detection. Each recording is written to a separate file, the name of which is specified in the prompt item definition in the script. The durations of the prerecording, recording and postrecording phase are specified within each prompt item.

3.3. Audio libraries and recordings

Java provides the powerful `javax.sound.sampled` API for audio processing. Unfortunately, this API is quite complicated to use, and it is not supported to its full extent on all platforms. Furthermore, high-quality audio equipment often provides features which are not accessible via a given platform’s implementation of `javax.sound`.

Hence, the `ipsk.audio` interface was developed at BAS. It is limited to the basic requirements of speech recordings, and it can be implemented using various audio APIs. SpeechRecorder comes with a standard `javax.sound` implementation of the audio interface. An implementation using ASIO drivers, which are standard for high quality studio audio hardware, is available on the BAS web site [29]. With the appropriate hardware, recordings with more than two channels can thus be performed.

Recordings in SpeechRecorder are either made to the local hard disk, or they are transferred via the WWW to a speech server.

Recording via the Internet requires that a server is configured to receive signal data from the client. SpeechRecorder stores the recorded signals in a temporary buffer and uploads the data. This upload is performed in a background process during the recording session, so that the recordings within a session can continue without having to wait for upload to complete.

To speed up the data transfer, the signal data is compressed using the lossless compression scheme flac (*free lossless audio compression*) [30]. For speech in a reasonably quiet environment, flac achieves compression rates of up to 50%.

3.4. Platform independence

SpeechRecorder was designed from scratch in a clean object-oriented design. It is written entirely in Java, and it requires version 1.4.2 or newer. As a standalone application it only requires the JRE on the machine. When downloaded from the WWW, the browser checks for the presence of an appropriate JRE; if none is found, it is downloaded and installed on the client (provided the necessary privileges are granted).

SpeechRecorder has been used successfully in a number of recording projects, e.g. the BITS synthesis corpus recordings (48 kHz stereo audio plus laryngograph signal) [31], on laptops in mobile environments, e.g. in the car, and in clinical environments.

3.5. User interface

Speaker and experimenter have different views on the recording data: a speaker needs to see only the prompt items and the recording control to tell him when to speak. The experimenter also needs to see a graphical signal display and level meter to check the signal quality and level adjustment. Furthermore, he needs to see the prompt item description and the list of all items of a recording session.

SpeechRecorder features a speaker and an experimenter view, and it supports and automatically recognizes multiple displays (fig. 4).

The user interface is localizable; localized versions exist for German, English, French, Polish and Russian.

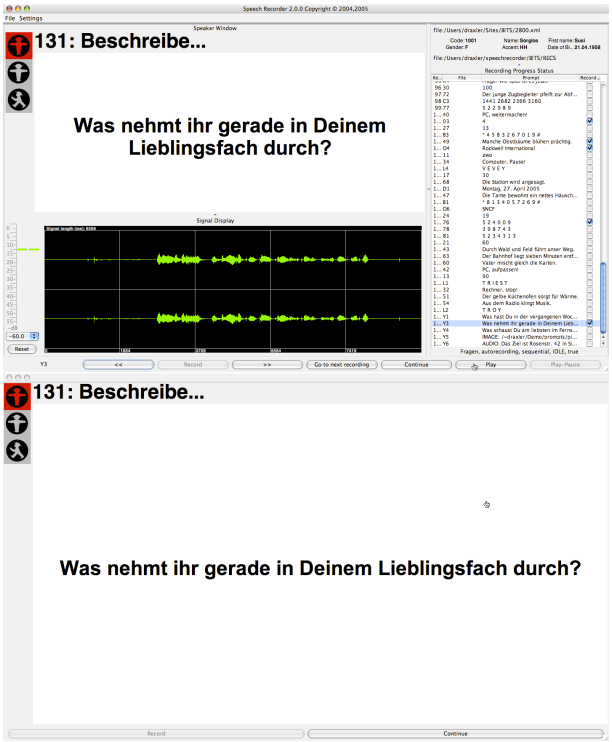

Figure 4: SpeechRecorder experimenter and speaker screen

4. WebTranscribe

WebTranscribe is an annotation framework [32]. As such, it implements the select-annotate-save cycle workflow:

1. *select* a signal file to annotate,
2. *annotate* the given signal, and then
3. *save* the annotation on the server.

The client requests the data necessary for the annotation from the server, and returns the annotation text to the server to be included in the database.

4.1. Annotation editors

WebTranscribe provides only the core functionality, i.e. data storage and access. The annotation functionality is provided via annotation editors which are implemented as plug-ins.

An annotation plug-in consists of a graphical user interface with an editor, usually a text or graph editor, and editing support buttons that perform frequently used annotation tasks, e.g. conversion from lower to upper case, or digit to string conversions.

The annotation editors register with the WebTranscribe framework, and they are notified when the underlying data model changes. Such a change may be the result of an editing action in the editor, or a signal selection in the graphical signal display. For a particular annotation task, multiple annotation editors may be used within the same window: for example, the annotation of SmartWeb recordings is performed using an editor for the orthographic transcript, and a second editor for an enhanced annotation.

The annotation plug-in also performs a lexical consistency check of the annotation text. Only if the text is formally correct can it be saved to the database.

In its default configuration, WebTranscribe consists of an info panel to display adminstrative data on the signal, a graphical signal display, an orthographic transcription editor, and a signal quality assessment panel (fig. 5). This configuration is performed by the system administrator on the server; for clients, no configuration is necessary.

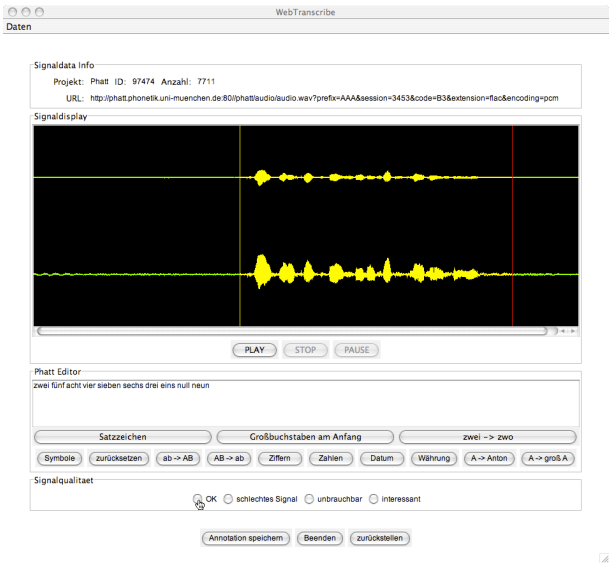

Figure 5: WebTranscribe configuration with orthographic transcription editor

Currently, annotation editors for orthographic SpeechDat-type transcriptions exist for German, English, French, Spanish, Russian and Hindi. Other annotation editors include an editor for the markup of the German SmartWeb project [33], and for the Münchner Verständlichkeitsprofil diagnostics tool [34].

4.2. `TranscriptionObject` Data structure

The main data structure for exchanging data between the server and the client is the `TranscriptionObject`. It consists of a reference to the signal, fields for administrative annotation data (annotator name, time stamps, etc.), a list of annotation items associated with the signal, and status information.

Initially, a signal that has not yet been annotated is transferred to the client with an empty default annotation. On the client, the annotation text is entered into the annotation field; any additional annotations of the same signal are appended to the list of existing annotations.

When a transcription object is returned to the server, the administrative data in the database is updated, and the annotations are inserted into the database. The next matching signal data and its associated annotations are then searched in the database and transferred to the server.

The status information serves specifies which operations to perform on the data: add or edit annotations, or view annotations (e.g. for validation), or quit the application.

4.3. Implementation

WebTranscribe is implemented in Java 1.4.2 or newer and is provided as a Java Web Start application. The server is an Apache Tomcat server accessing a standard relational database

system.

5. Ph@ttSessionz

Ph@ttSessionz (Phonetics at teenage-talk) is a project for recording 1000 adolescent speakers to create a speech database for speech technology development. The Ph@ttSessionz recordings are performed over the Internet in public schools all over Germany using the web application developed at BAS (fig. 6)[1] [35, 36].

5.1. Database contents

The Ph@ttSessionz database is a superset of the German SpeechDat telephone network speech database [37] and the RVG database [38]. The material recorded consists of isolated digits, formatted digit strings, e.g. telephone and credit card numbers, numbers, money amounts, date and time expressions, command phrases for PC operation, directory assistance names, e.g. person, city and company names, and phonetically rich words and sentences. Furthermore, speakers are asked to respond to a few simple questions, e.g. "What message do you leave on the answering machine of a friend?".

A recording session consists of three sections with a total of 125 items. The first section serves to acquaint the speaker with the software, and it allows the experimenter to adjust the level settings. The main section presents the items in random order, and the software progresses automatically through the recording script. The speaker may terminate a recording item by clicking on the appropriate button.

In the final section, open questions are asked. These questions are designed to elicit colloquial speech with a regional accent. The questions address topics the speakers are familiar with, e.g. their favorite school subject, their last holidays or some other interesting or funny event. By the time the speaker reaches this section, he is familiar with the recording procedure, and the speaking style is quite relaxed. The maximum recording duration for the answers is 60 seconds.

5.2. Demographic coverage

Ph@ttSessionz is designed to cover all major dialect areas of Germany (fig. 6) [39]. The speakers are between 13 and 18 years old, and the database is balanced by sex.

The following data is recorded for every speaker: age at the time of recording, sex, size, weight, smoking habits, presence of dental braces or piercings, mother tongue, federal state where he entered school, and native language of mother and father.

The educational level of all speakers is secondary high school (*Gymnasium*).

5.3. Recording equipment

To achieve a consistent signal quality, all recordings are performed with a standard recording equipment consisting of a Beyerdynamic opus 54 headset microphone, an Audio Technica 3031 desktop condenser microphone, and an M-Audio Mobile Pre USB analog/digital converter (fig. 7).

The recording equipment is packaged into cases to be sent to participating schools. 9 such cases were set up, 8 of them were sent to recording locations, one remained in our lab for testing purposes.

[1] Ph@ttSessionz is funded by the German Ministry of Research and Technology under grant no. 01IVB01 (BITS)

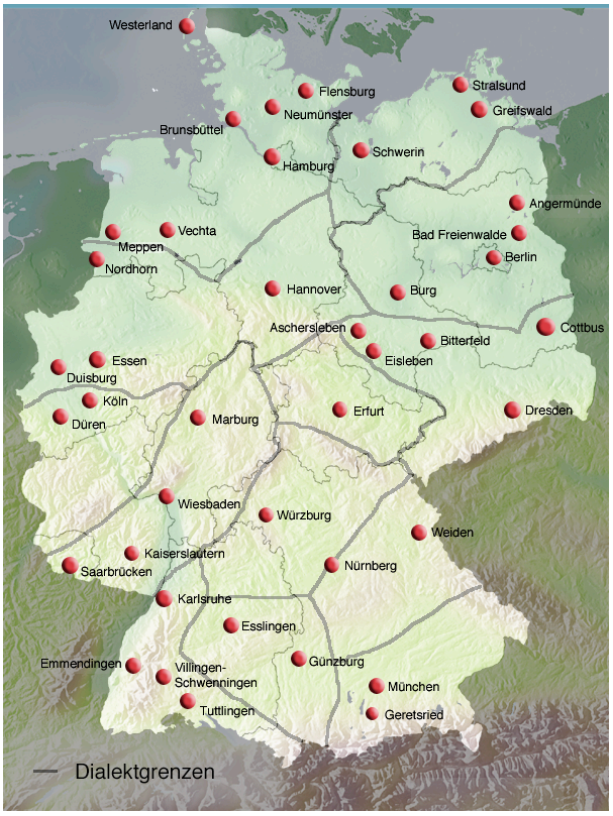

Figure 6: Ph@ttSessions recording locations and dialect regions

The sample rate is 22.05 kHz with 16 bit linear quantization and two channels, resulting in a data rate of 705.6 kBit/s.

5.4. Recruiting schools and speakers

In a first recruitment phase, high schools in larger cities within the dialect regions were asked to participate in the Ph@ttSessionz project. Initially we requested each school to record 50 speakers, but this was quicky reduced to 30, and we offered the school 200 Euro for the recordings.

Once a school agreed to participate, we scheduled the recording period and sent the school preparatory material, i.e. information leaflets for parents and pupils, registration forms, and a contract. The school named a person responsible for the recordings – this was often a teacher, or head of a special interest group, e.g. the Internet group.

This person was responsible for recruiting pupils for the recordings. Some schools allowed pupils to take part in the recordings instead of regular lessons – surely an interesting alternative for some –, while at other schools recordings took place outside the regular lessons.

In the beginning it was very difficult to get schools to participate in Ph@ttSessionz. However, once the first schools had participated, we had showcases to present. Our team members responsible for recruiting schools developed a guideline for telephone recruitment of schools which proved to be useful in standardizing communications with the schools.

It was particularly useful to contact local newspapers in parallel to the schools – the newspapers were interested in reporting about a high tech project, and the schools were interested in get-

Figure 7: Recording session with speaker and experimenter in Erfurt

ting press coverage. Furthermore, we made the experience that schools in smaller cities were more interested in participating than those in larger cities.

5.5. Recording procedure

Prior to any production recordings, the experimenter at the school had to test the recording setup. Only when this test was passed successfully did recordings with pupils begin.

The experimenter logs in to the Ph@ttSessionz web application. He then opens a new session via the menu and enters the speaker data, submits it to the server and starts the SpeechRecorder software.

SpeechRecorder fetches the prompt sheet for the current recording session from the server and checks whether any items from the current session have been recorded already. Once the speaker is comfortable with the procedure and the recording levels have been adjusted on the analog/digital converter, the experimenter leaves the room and the recording proceeds automatically without supervision.

A recording session takes approx. 25 to 30 minutes under normal conditions. When all items have been recorded the session is closed. The software continues uploading the signal data until all data is transferred.

5.6. Infrastructure at BAS

For Ph@ttSessionz, a web application has been developed through which all services are accessed [40]. This web application distinguishes four classes of users:

1. the *general public* may view the public pages only, i.e. general information, press articles, a video demo of a recording session, etc.
2. *experimenters* at a recording location may perform the sine test, start recording sessions, and view the recording statistics for their site
3. *transcribers* at BAS perform the orthographic transcription of the recorded signal files using the WebTranscribe application
4. *administrators* monitor the operation of the server, update the server contents, e.g. with new scripts, compute global statistics and select the sessions to be annotated.

All users except the general public must log into the web application to access the services. A user can view who else is logged in, and may exchange messages with the adminstrator.

The web application allows multi-user access. This is most useful for online monitoring of recordings, especially at locations where the first recordings are just starting. The BAS administrator can immediately check incoming signals for their quality (signal level, noise, etc.) and contact the remote experimenter in case of problems.

The Ph@ttSessionz web application is implemented using Java Server Pages technology on an Apache Tomcat server using the PostgreSQL relational database system on a standard Linux PC.

5.7. Problems

To test the technology, field tests were performed in two schools in Munich (Oskar-von-Miller-Gymnasium OvMG) and Geretsried near Munich (Gymnasium Geretsried GG). During these field tests, a number of technical problems were encountered [41].

5.7.1. Corrupted signals

At OvMG, signal files showed dropped frames at irregular intervals. We suspected that either the processing power of the recording PC was not sufficient, or that there was some incompatibility between the operating system, the Java runtime engine, the USB drivers and the audio hardware.

We thus set up a test machine at our lab that consisted of an old 400 MHz PC with a slow network card (ISDN) and a modern Windows XP operating system. However, even this weak machine was sufficiently powerful to perform the recordings without problems.

We then tested different combinations of Windows operating systems, Java versions and USB drivers, i.e. either the built-in default USB audio drivers, or the M-Audio drivers. Again, the result was inconclusive.

To at least detect dropped frames we thus developed the sine test, which had to be performed by every recording site. In this test, a sine wave produced by the PC was recorded via the USB analog/digital converter. At our lab, we examined the recorded signal using a sonagram to visually detect corrupted signals.

Furthermore, we recommended that all sites update their operating system with the latest software patches which we included on a CD, together with the M-Audio USB drivers.

5.7.2. Slow data transfer

The data rate of 705.6 kBit/s is higher than the upload speed of normal DSL connections (128-384 kBit/s). Thus, data transfer took longer than recording the signals. In the first version of SpeechRecorder, a new item could only be recorded when the signal data of the previous item had been uploaded to the server, leading to recording sessions that were much longer than planned.

We modified the software to buffer the recorded signals and to proceed with the recording session while data transfer was running in the background. We hoped that the time between sessions would be sufficient to upload the data of the previous session. For some schools, this worked fine, but other schools had a very tight recording schedule, and they could not wait between sessions.

In order to avoid having to wait, some experimenters simply started a new session. Although the web application did

not allow starting a new session while the upload of the previous session was still under way, some experimenters found a workaround to avoid this restriction: they simply used the ”back” button to return to the speaker form, entered new speaker data and started a new session. However, SpeechRecorder had only one buffer, and as a consequence the new recordings would overwrite the contents of the old buffer, leading to corrupt recording sessions.

SpeechRecorder was thus modified again to assign a separate buffer to each recording session. New sessions could be started any time, so that recordings would now proceed smoothly. At the end of a recording day, the experimenter had to delay system shutdown until all data was uploaded.

Other factors affecting the data transfer rate were firewalls, which very often were not administered by the schools but by external companies (e.g. at OvMG and GG). If the firewall caused significant delays, and if its settings could not be changed by the school, we aborted the recordings at this site.

Finally, early versions of Java 1.5 sent data packets repeatedly through firewalls, each time with different certificates. Fortunately, this problem disappeared with later releases of Java 1.5.

5.8. Current status

Production recordings started in April 2005. By end of March 2006 we have reached 684 speakers from 33 schools (exluding the field tests at OvMG and GG). A total of 87.800 items have been recorded so far.

Transcriptions have started in January 2006, and by end of March we have transcribed more than 16.000 utterances, comprising more than 11 hours of speech.

The transcription of the short read items is quick – very often one needs to listen to the signal only once, and then automatically convert the prompt text into its string representation using an editing button, e.g. from ”1 2 3” to ”eins zwei drei”, select the appropriate signal fragment, and enter an assessment of the signal quality.

Longer items, such as long digit strings, sentences or the spontaneous speech, are more difficult to transcribe. They often contain hesitations such as ”uh”, speaker noises such as laughter, lip smacks or breathing, or mispronunciations.

Table 1 gives preliminary results for the annotation of the Ph@ttSessionz recordings. Note that for the read items, i.e. sentences, spellings, digit strings and isolated digits, annotation times of more than 10 minutes were discarded from these calculations because they are artefacts: the annotators did not close the annotation session when they left their terminal for a break.

type	code	segment length	annotation time
spontaneous response	Y1-Y3	00:11.35	02:26.77
read sentences	43-72	00:02.74	00:17.93
spelling	L1-L9	00:04.69	00:18.54
credit card no.	C1-C3	00:07.25	00:28.49
10 digit string	B1-B3	00:05.36	00:21.34
single digit	01-10	00:00.73	00:09:70

Table 1: Average segment lengths and annotation times for a subset of the Ph@ttSessionz speech database

6. Conclusions and Outlook

We have shown that speech databases can be viewed as a service, accessible via the WWW. The web and its technologies, namely Java Web Start and web applications, are mature, stable and sufficiently powerful for the demands of recording, annotating and exploiting speech databases.

Recording speech via the WWW allows geographically distributed high-bandwidth recordings, and thus is a viable alternative to inviting speakers to come to a lab, or to visit speakers with a team of experimenters. In fact, with web-based recordings entirely new speaker populations can be accessed, e.g. patients in hospitals, children in pre-school, office workers in their normal working environment, or even people at home – all that is needed is a reasonably fast Internet connection.

Web-based annotation also offers unique opportunities – flexible and simultanous access to shared data, immediate update of the central resources, scaleable distribution of workload and platform independence.

Current work at BAS involves a redesign of the SpeechRecorder recording script definition language to allow for more flexibility and improved ease of use, the implementation of a recordings script editor and improved logging. For WebTranscribe, new editor plug-ins are being developed. The internal data model of annotations will be extended to allow true multi-level annotations, either time-aligned or with symbolic references.

7. Acknowledgments

Many people have contributed to the work described here. I would like to thank Klaus Jänsch for his excellent programming work, Angela Baumann and Tania Ellbogen for their recruitment of schools, and the students who are busy transcribing the Ph@ttSessionz speech data.

8. References

[1] Huckvale M., Brookes D., Dworkin L., Johnson M., Pearce D., Whitaker L., ”The SPAR Speech Filing System”, 1987, Edinburgh

[2] Boersma P., Weenink D., ”Praat, a System for doing Phonetics by Computer”, Institute of Phonetic Sciences of the University of Amsterdam, TR 132, 1996, Amsterdam.

[3] Cassidy S., Harrington J., ”EMU: an Enhanced Speech Data Management System”, Proceedings of SST96, 1996, Adelaide.

[4] Barras C., Geoffrois E., Wu Z., Liberman M., ”Transcriber: a Free Tool for Segmenting, Labeling and Transcribing Speech”, Proceedings of LREC, 1998, Granada.

[5] Sjölander K., Beskow J., Gustafson J., Lewin E., Carlson R., ”WaveSurfer – an open source speech tool”, Proceedings of ICSLP, 2000, Bejing.

[6] Draxler Chr., ”WWWTranscribe – a Modular Transcription System based on the WWW” Proc. of Eurospeech, 1997, Rhodes.

[7] Draxler Chr., ”WWWSigTranscribe – a Java Extension of the WWWTranscribe Toolbox” Proc. of LREC, 1998, Granada.

[8] McKelvie D., Isard A., Mengel A., Möller A., Grosse M., Klein M., ”The MATE Workbench – an annotation tool for XML coded speech corpora”, Speech Communication 1-2, pp. 97-112, 2001.

[9] Ma X., Lee H., Bird S., Maeda K., ”Models and Tools for Collaborative Annotation”, Proc. of LREC, 2002, Gran Canaria.

[10] Fielding R., Gettys J., Mogul J., Frystyk Nielsen, H., Masinter L., Leach P., Berners-Lee, T., ”Hypertext Transfer Protocol – HTTP/1.1”, W3C RFC 2616, 1999.

[11] Berners-Lee T., Fielding R., Masinter J., ”Uniform Resource Identifiers (URI): Generic Syntax”, W3C Proposed Recommendation, 1998.

[12] Ragett I., Le Hors A., Jacobs I., ”HTML 4.0 Specification”, W3C TR REC-html40, 1998.

[13] Pemberton S., et al., ”XHTML 1.0: The Extensible Hypertext Markup Language – A Reformulation of HTML 4.0 in XML 1.0”, W3C Proposed Recommendation, 2000.

[14] Bray T., Paoli J., Sperberg-McQueen C., Maler E., ”Extensible Markup Language XML 1.0”, W3C Recommendation, 1998.

[15] Esling J., ”Computer Coding of the IPA: Supplementary Report”, Journal of the IPA, Vol. 30 No. 1, 1990.

[16] Gibbon D., Moore R., Winski R., eds., ”Handbook of Standards and Resources for Spoken Language Systems”, Mouton de Gruyter, 1997, Berlin.

[17] Oostdijk N., ”Meta-Data in the Spoken Dutch Corpus Project”, MPI ISLE Report, 2000.

[18] Lemnitzer L., Zinsmeister H., ”Korpuslinguistik – eine Einführung”, Narr Francke Attempto Verlag, 2006, Tübingen.

[19] Aston G., Burnard L., ”The BNC Handbook: Exploring the British National Corpus with SARA”, Edinburgh University Press, 1998.

[20] Cunningham H., Maynard D., Bontcheva K., Tablan V., ’GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications, Proc. of 40th ACL Meeting, 2002.

[21] Oostdijk N., Broeder D., ”The Spoken Dutch Corpus and its Exploitation Environment”, 2003.

[22] Cosmas II, `www.ids-mannheim.de`

[23] TIGERCorpus, `www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/`

[24] Broeder D., Brugman H., Russel A., Wittenburg P., ”Browsable Corpus: accessing linguistic resources the easy way”, LREC workshop, Athens, 2000.

[25] European Language Resources Association, `www.elra.info`

[26] Linguistic Data Consortium, `www.ldc.upenn.edu`

[27] Cunningham H., Tablan V., Bontcheva K., Dimitrov M., ’Language Engineering Tools for collaborative corpus annotation, Proceedings of Corpus Linguistics, 2003, Lancaster

[28] Draxler Chr., Jänsch K., ”SpeechRecorder – A Universal Platform Independent Multi-Channel Audio Recording Software”, Proc. of LREC, 2004, Lisbon.

[29] Bavarian Archive for Speech Signals, `www.bas.uni-muenchen.de`.

[30] Free Lossless Audio Compression, `flac.sourceforge.net`

[31] Ellbogen T., Schiel F., Steffen A., ”The BITS Speech Synthesis Corpus for German”, Proc. of LREC, 2004, Lisbon.

[32] Draxler Chr., ”WebTranscribe – An Extensible Web-Based Speech Annotation Framework”, Proc. of TSD 2005, Lecture Notes in Computer Science, pp. 61-68, Springer Verlag, 2005, Berlin.

[33] Wahlster W., SmartWeb: Mobile applications of the semantic web. Informatik, 2004

[34] Ziegler W., Hartmann E., ”Das Münchner Verständlichkeitsprofil (MVP) – Untersuchungen zur Reliabilität und Validität, Nervenarzt, No. 64, pp. 653-658, 1993.

[35] Steffen A., Draxler Chr., Baumann A., Schmidt S., ”Ph@ttSessionz: Aufbau einer Datenbank mit Jugendsprache”, Proc. of DAGA, 2005, Munich.

[36] Draxler Chr., Steffen A., ”Ph@ttSessionz: Recording 1000 Adolescent Speakers in Schools in Germany”, Proc. of Eurospeech, 2005, Lisbon.

[37] Höge H., et al., ”SpeechDat Multilingual Speech Databases for Teleservices: Across the Finish Line”, Proc. of Eurospeech, 1999, Budapest.

[38] Burger S., Schiel F., ”RVG-1 – A Database for Regional Variants of Contemporary German”, Proc. of LREC, 1998, Granada.

[39] Hollmach, U. ”Untersuchungen zur Kodifizierung der Standardaussprache in Deutschland”, Habilitationsschrift, Universität Halle, 2003.

[40] Ph@ttSessionz web site at BAS `www.phonetik.uni-muenchen.de/phatt`.

[41] Draxler Chr., Jänsch K., ”Speech Recordings in Public Schools in Germany – the Perfect Showcase for Web-Based Recordings and Annotation”, Proc. of LREC, 2006, Genova.