002

TR-IT-0312

# Client-Server Speech Translation System ATR-MATRIX

グルーン ライナー
Rainer Gruhn

シンガー ハラルド
Harald Singer

September 6, 1999

We describe the implementation of a client-server based speech translation system. To minimize the bandwidth problem, input speech data is preprocessed and compressed and then sent to the recognition server. For synthesizing the translated utterance we implemented a variety of approaches requiring between 256 kbps for high quality speech down to less than 1 kBps for unit information. The client was implemented for Windows95/98, Linux and OSF1. The system was also tested using a 9.6 kBps mobile telephone data connection. This report is accessible for ITL members via `/home/singer/tex/TR-IT-MATRIX/`.

# Contents

## 1 Introduction

The speech-to-speech translation system ATR-MATRIX [Reaves et al., 1998, Takezawa et al., 1998] is a research prototype with high recognition accuracy. It was designed with a highly modular structure, consisting of a speech recognition subsystem (ATRSPREC), a speech synthesis subsystem (CHATR) and a main controller (see Figure 1).

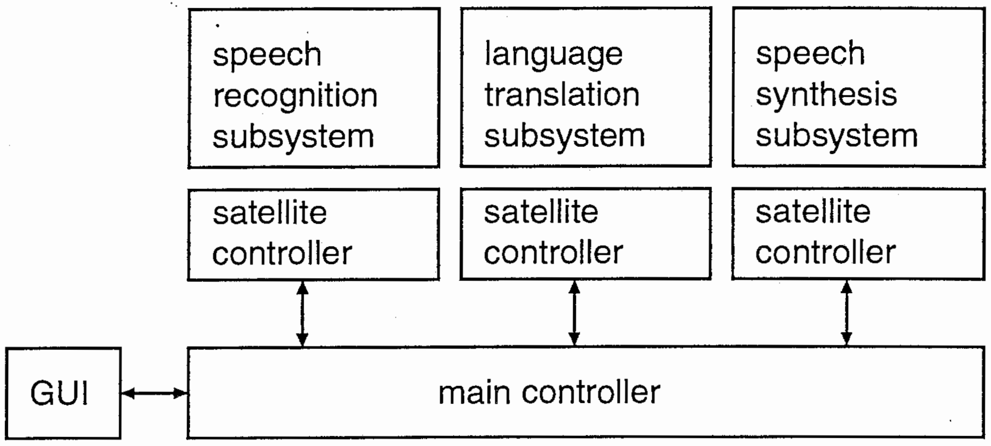
speech recognition subsystem
language translation subsystem
speech synthesis subsystem
satellite controller
satellite controller
satellite controller
GUI
main controller

Figure 1: ATR-MATRIX system architecture

This stand-alone version of ATR-MATRIX has previously been evaluated for naive users [Sugaya et al., 1999]. It was also successfully demonstrated in the C-Star international experiment on July 22nd, 1999 [C-Star, 1999].

Still, there are problems that limit its applicability:

- Running ATR-MATRIX requires complex equipment, making it difficult and expensive to demonstrate in a different place than the ATR laboratories. There is also a certain risk of computers getting damaged during transport. After transport, the equipment must be set up again. To sum up, demonstrating ATR-MATRIX is a risky, costly and time-consuming matter. Using it, i.e. going abroad and communicating with other people while carrying all the equipment is an impossible task.
- Installing ATR-MATRIX is a difficult task, requiring a lot of time and occasional help from the programmers. Also running it is not easy, a certain order of commands has to be given to start it and several details have to be checked upon shutdown, otherwise a restart is impossible.
- Among common users, the most popular operating system is Microsoft Windows. Programs that run under other operating systems can reach only a limited number of people, which means that ATR-MATRIX has to run under Windows to be widely accepted. But porting the whole ATR-MATRIX system from Unix to Windows is a difficult task. Also, Windows is not a very stable operating system.
- Even if ATR-MATRIX were a popular product, it could not easily be distributed. Some parts of ATR-MATRIX make use of commercial program libraries, that cannot be copied and given to customers without special licensing arrangements, e.g. the RogueWave class library.

These problems, in particular the goal to "get ATR-MATRIX out of the laboratory into the real world" motivated us to introduce a client-server system architecture for ATR-MATRIX. Dividing ATR-MATRIX into a user front end and a server main part solves most of the problems listed above or at least hides them from users. The system becomes portable and easy to install and use.

Section 2 describes the advantages and problems of client-server architecture. Section 3 is a comprehensive guide through a sample session. Section 4 shortly explains the basics of a telephone based ATR-MATRIX system.

The appendices give detailed technical information about how to install the client and server software, describes the programs added to ATR-MATRIX for client-server architecture, answers possibly occuring questions and finally lists used socket port numbers and data types.

## 2 Client-Server Architecture

ATR-MATRIX [Reaves et al., 1999] consists of five main parts:

- ATRSPREC for speech recognition,
- TDMT for translation,
- CHATR for speech synthesis,
- MatrixGui as user interface,
- the ATR-MATRIX controllers who coordinate the four modules above and maintain connections to other speech recognition and translation systems.

The purpose of a client-server architecture is to get all complex parts on the server. For a speech recognition, translation and synthesis system, this is the heavy computation requiring speech recognition process and the speech synthesis with its large wave file database. Only audio recording (AD), audio display (DA) and the GUI should run on the client. Figure 2 shows the basic structure of a client-server ATR-MATRIX.

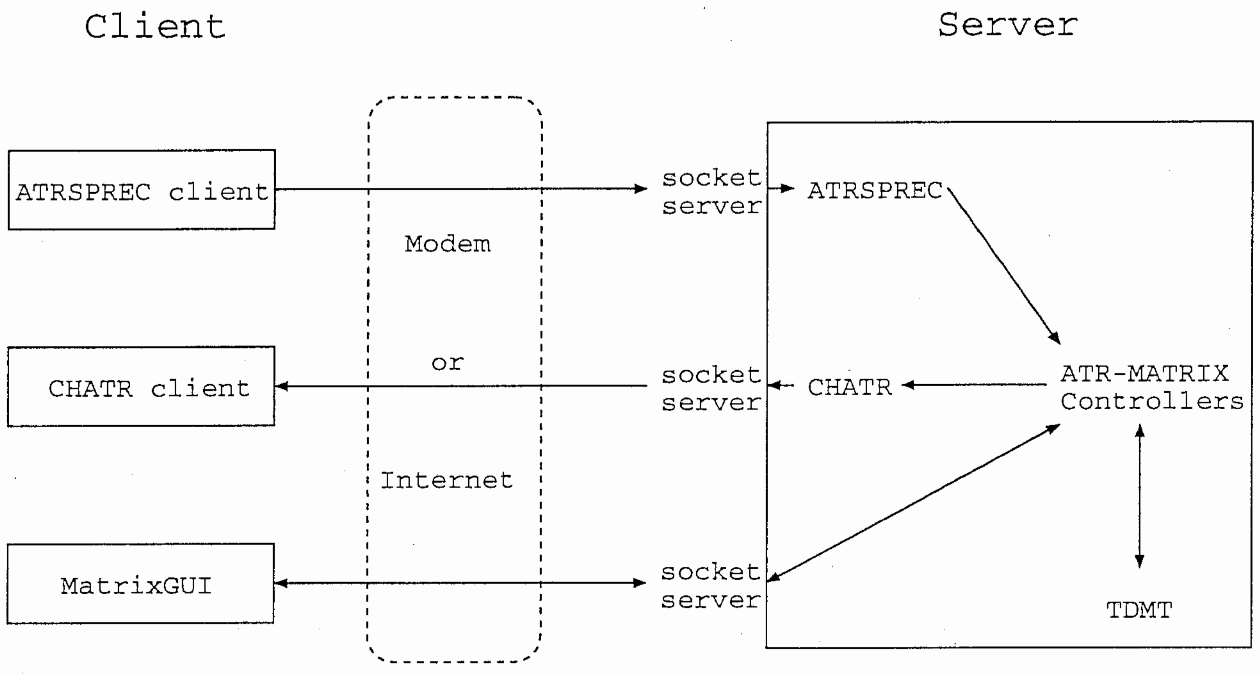

Figure 2: Structural overview of a client-server system.

### 2.1 The Bandwidth Bottleneck

There is a limiting factor between client and server: the connection bandwith (dotted line in Figure 2). To be able to use high quality speech data for recognition and as synthesized answer, 256 kBps of data have to be transfered for each input and output. Such a bandwidth requirement would limit the use of client server to local networks, as no ISDN modem connections can offer 512 kBps bandwidth. To be able to use a client from anywhere, bandwidths have to be restricted to much lower values, e.g. 56 kBps for a modem or 9.6 kBps for a cellular phone modem (see Table 1).

For this reason, we have to look at the three connections that are necessary between client and server:

- between ATRSPREC client and recognition server
- between the speech synthesis server and the client's audio device
- for the GUI

Table 1: Currently available data transfer bandwidths

| communication system | bit rate |
|---|---|
| ISDN-modem | 128 kBps (2 chnl) |
| modem on analog phone line | 14.4 - 56 kBps |
| modem on cellular phone | 9.6 - 33 kBps |
| internet | ??? |

The connection between GUI and server does not require much bandwidth, so there is no need for compression.

To reduce the data stream for the audio input, compressing the speech e.g. by downsampling from high quality speech (16 bit linear data, 16 kHz sampling rate) to telephone quality ($\mu$law compression, 8 kHz) is not a good option as it reduces the recognition rate. The solution is not to transfer the speech, but to extract the necessary information from the speech and transfer only that. To archieve this, the client does not only record the speech signal, but also performs endpoint detection and preprocessing down to the cepstral parameters. These parameters are further compressed and transferred in bitvector form, with a total bandwidth of e.g. 5.6 kBps. A full description of the compression can be found in [Gruhn and Singer, 1998] and [Gruhn et al., 1999]. The preprocessing and compression does not require a fast CPU. It was tested on a Pentium 133 processor under the Windows 95 operating system and runs in realtime. A test on a 486 processor laptop had to be cancelled because of hardware defects, but first results allow the estimation that a 486 processor cannot fully provide sufficient computational power.

The other high bandwidth data stream is the synthesized translation from CHATR. The problem is similar: the high quality audio data requires too much bandwidth. Compression is available, the toll quality $\mu$law compression and the cellular phone standard GSM are well-known examples. But lower bandwidth has to be paid with a loss in audio quality. An alternative is to split up the speech synthesis process into unit selection and wave concatenation. The server does the unit selection: the determination which of the wave file chunks in the speech database are used in which order. The client does the wave form concatenation: given the list of wave files to use, it loads them from the data base and plays them. This system uses only minimal bandwidth, but requires the CHATR database for the desired speakers to be installed on the client's computer[1].

The necessary hard disk space and fast disk access may be unavailable (see Appendix A for details). Therefore, the ATR-MATRIX client-server system gives the client the choice to either receive a concatenated (and optionally compressed) speech signal or only wave unit information.

## 2.2 Client Audio Device

Independence from computer operating systems is an important advantage of client-server architecture. A big problem is the audio device in the client computer, especially the question of full duplex capability. Modern high quality sound devices support full duplex, that is the ability to play and record at the same time. But many computers have old or cheap sound devices that are half duplex, i.e. they can only either record or play at the same time. Other sound devices seem to support full duplex, but mix input and output signals[2].

To avoid such half-duplex problems, the client software must schedule the audio device accesses. In particular, audio recording must be suspended while a synthesized wave is played. The drawback of this approach is, that input speech is "mercilessly thrown away" if synthesized speech arrives on the client, i.e. DA takes precedence over AD. The GUI and the CHATR client programs do this by sending "SLEEP" and "AWAKE" commands to the ATRSPREC client.

The complete client-server ATR-MATRIX is shown in Figure 3.

## 2.3 Resource Management

A client-server architecture can also enhance machine use efficiency. The direct implementation of the system described in Figure 3 would be a one-client one-server structure, with each server being fixed to one client only. Client-server allows a system with a pool of servers being accessed by to clients on demand. Also,

[1]this requires hard disk space in the order of 100 MB per speaker

[2]even on a highend PentiumII-300 Gateway Solo we found that the output is directly fed back into the AD input

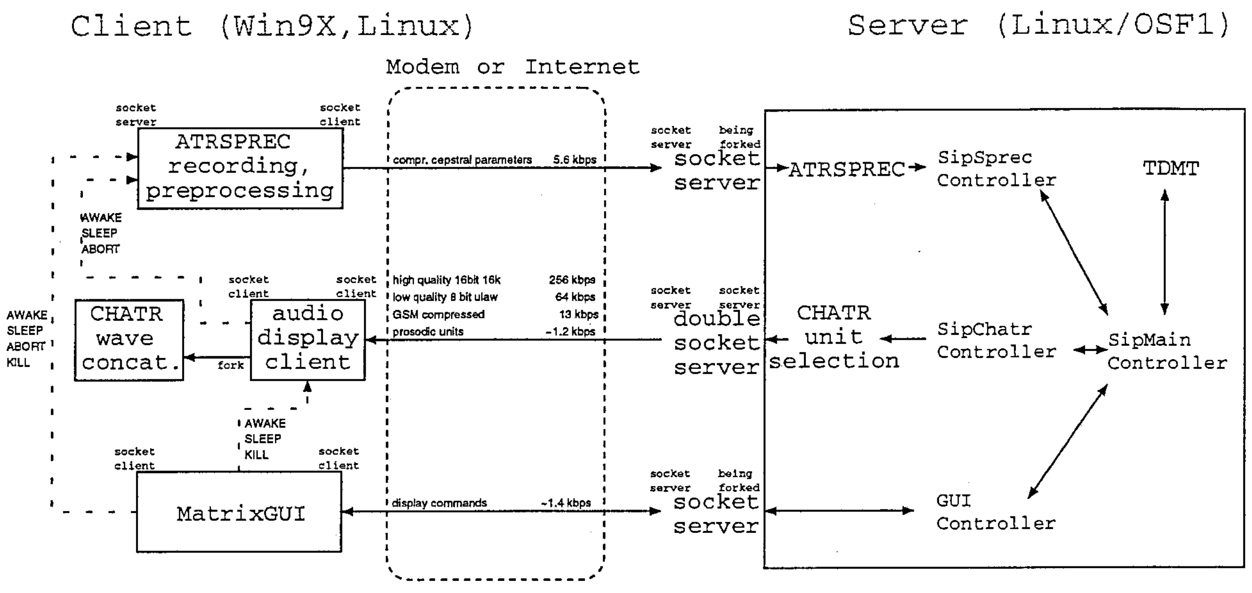

Figure 3: System overview.

various server types can be supported, e.g. with different input languages or even different preprocessing conditions.

In addition to the servers and clients, a resource manager is needed, waiting on a well known port and host for login requests, holding knowledge about server properties. If a client sends a login request, it tells the resource manager some client properties and which kind of server is required (e.g. which language will be the input language). The resource manager then checks in the list of known servers, which servers match the request and are unused. If a matching server is found, the resource manager marks it as used and the client receives a SIP-format message [Handley et al., 1997] containing the IP address of a free server. Otherwise, the client gets a "busy" error message.

All data communication is done between client and the assigned ATR-MATRIX server directly, except for the session end, which the resource manager is told of.

The administration of servers and their distribution to clients is shown in Figure 4.

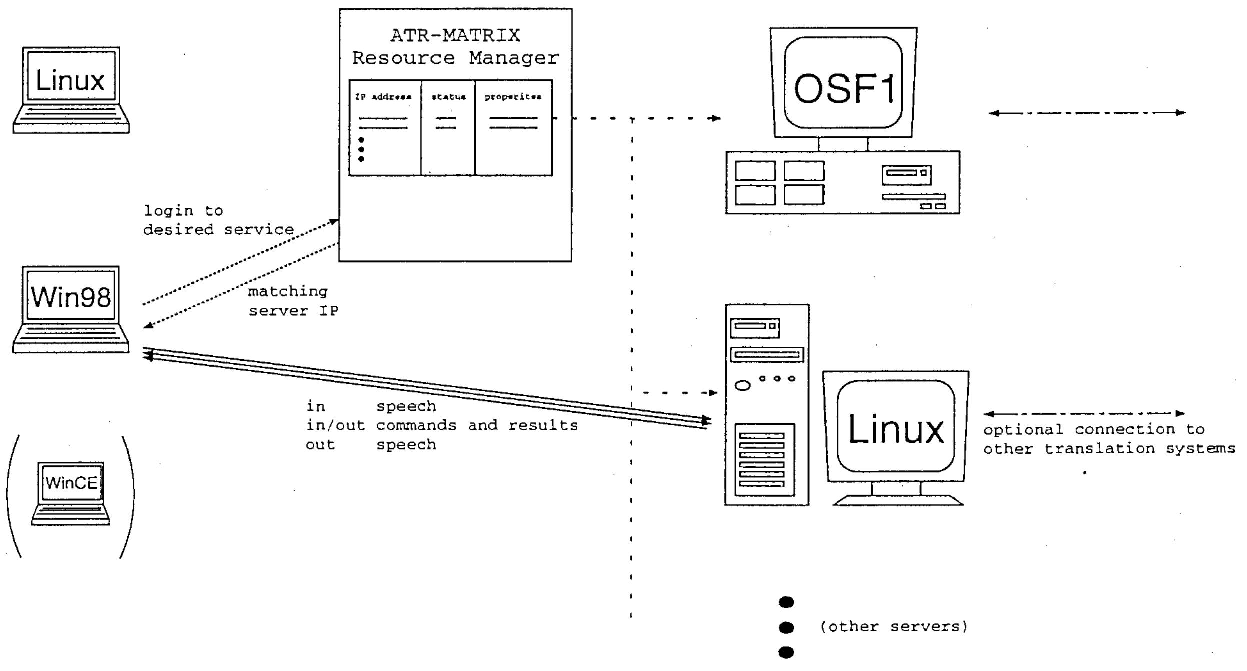

Figure 4: Resource management overview.

## 2.4 Communication with Other Systems

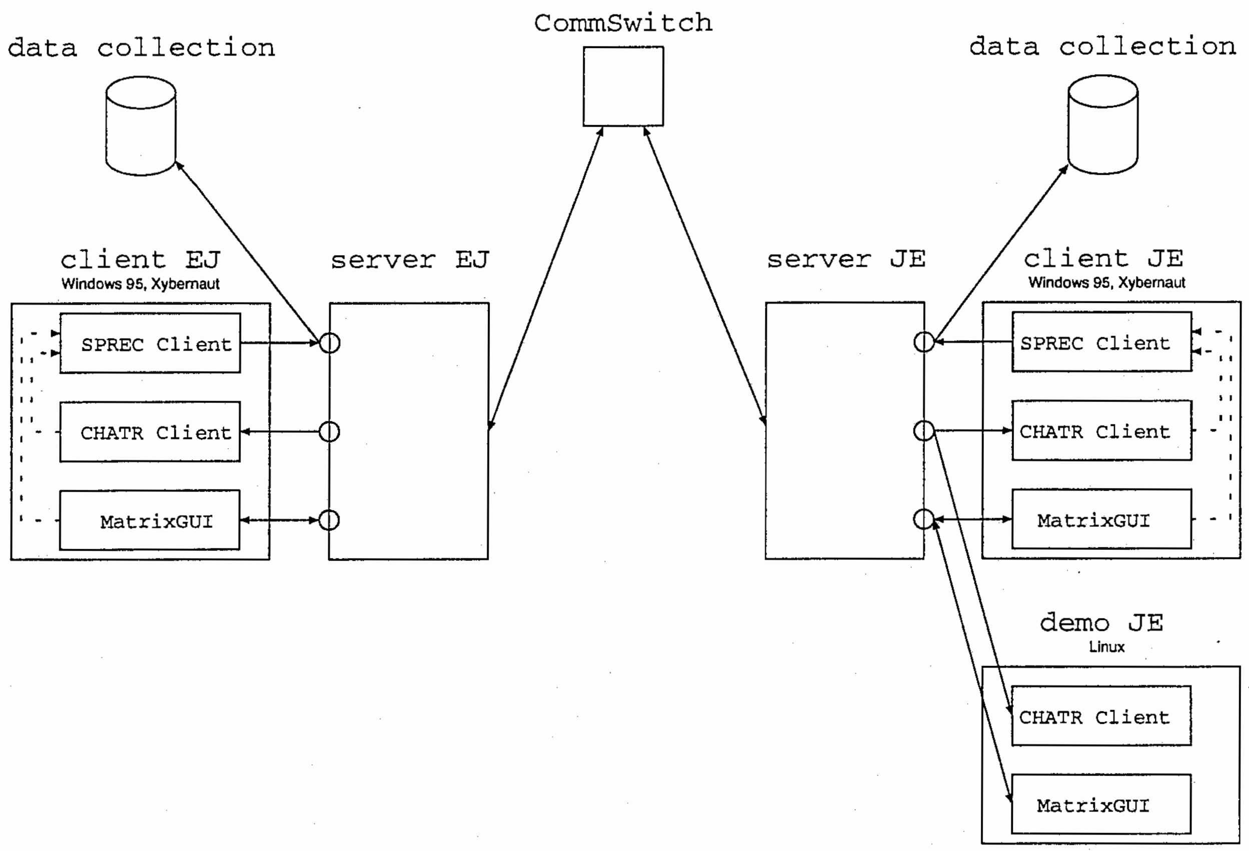

Figure 5: Example of an English-Japanese ATR-MATRIX connected to a Japanese-English ATR-MATRIX via a communication switch used in CSTAR II.

To test the viability of a cross language translation system, evaluation tests between naive Japanese users and trained English users were performed [Sugaya et al., 1999]. These experiments were performed on workstations.

The client-server architecture can also accomodate this paradigm and an example is shown in Figure 5. One advantage of the client-server architecture is, that only the client portion of ATR-MATRIX has to be ported to the target machine: here we implemented the client on a "wearable" Xybernaut, that runs under Windows95. This application shows an additional advantage of client-server: the ability of "listening in" to a conversation by copying the server output (recognition/translation results and synthesized translated speech) to another machine. For a demo of the wearable system this is particularly important as directly copying the screen of the wearable is slow, unstable (Windows) and requires special hard- and software. Figure 6 shows a picture taken at the CStar II experiment [C-Star, 1999] which was published in [Yomiuri Shimbun, 1999]: Two people using ATR-MATRIX clients installed on wearable computers for a dialog in Japanese and English. In the background of the picture a projection of the GUI window can be found which both dialog partners also see in their personal little screen before their left eyes. On the projection the audience can see the recognition and translation results. The audio data is available accordingly.

Figure 6: A picture taken at the CStar II experiment on July 22nd 1999.

# 3 A Sample Session

This chapter gives a comprehensive description of a typical client-server session. Additional details about the programs, such as command line options or RPCs, can be found in Appendix B .

## 3.1 Assumptions

There are some prerequesites that have to be met before the actual session starts.

- At least one server is running and the resource manager knows about it. If this condition is not met, the client will get the error message "busy" when trying to start a session.
- `matrixlogon.py`, the resource manager, is running on `133.186.35.196` and waiting on port `42000`. If a connection to this port is impossible, the client will get the error message "server not ready" when trying to connect.
- The client should have the newest client software installed on his computer. Older versions will work, but their use is discouraged as some features may be unavailable. Their behaviour may also be different from the behaviour described in this chapter. Appendix A gives installation information.

## 3.2 Login

This section explains the startup procedure. A graphical overview is also given in Figure 7.

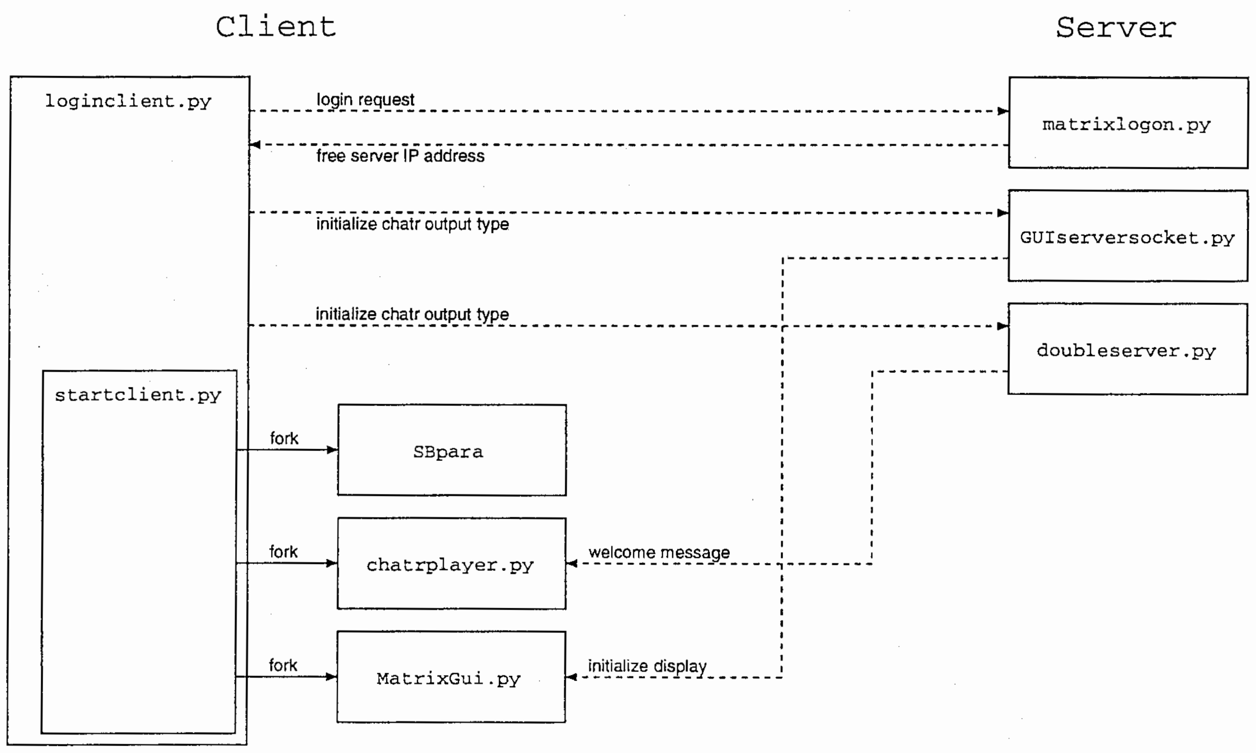

Figure 7: Overview over the messages sent and proceses forked to start a ATR-MATRIX client.

To open a session, the user starts the program `loginclient.py` or, under Windows, doubleclicks on the icon startmatrix. The behaviour of this program may be altered, e.g. to start a session with English as input language instead of the default Japanese. Appendix B.1 explains the command line options.

`loginclient.py` opens a socket connection to `matrixlogon.py` and sends a login message in SIP-format [Handley and Jacobson, 1998, Handley et al., 1997]. Here is an example of the typical login message:

```
INVITE\r\n
Content-Length:53\r\n
\r\n
a=input:j\r\n
a=clienttype:computer\r\n
a=matrix:2.4\r\n
v=2\r\n
```

The message consists of two parts which are separated by an empty line. First comes the header containing the message type and the body content length. The message body contains (in the order of the example above):

- input language (English or Japanese),
- type of client (a computer or a telephone connection),
- version of ATR-MATRIX the client expects (this information is currently not used),
- login message version (to be able to detect old client software versions).

`matrixlogon.py` reads the request, parses it and searches its database for a free server that matches the request. It returns either an error message (e.g. `451 Busy`) or an OK message telling the client which server to connect to:

```
OK\r\n
Content-Length:18\r\n
\r\n
c=133.186.34.204\r\n
```

The next step is server initialization: the ATR-MATRIX server supports various output types for CHATR, wavefile concatenation information (so-called chatr "units") and audio in GSM-, ulaw- or linear 16kHz/16bit format. The data fastest to transmit are the chatr units. But this can only be used if the wave database and the wave file concatenation program (`concat`) are installed on the client. To tell the server which audio type to send, `loginclient.py` sends messages to the server programs `doubleserver.py` and `GUIserversocket.py`. `loginclient.py` checks for the existence of `concat`, if it is found, the server is set to send units, otherwise audio in ulaw format. The initialization message for `doubleserver.py` is a RPC with the command `chatrtext`, to the `GUIsocketserver` the following string is sent (as one line):

```
chatrcommand (Audio Command "cat /tmp/chatrjunk.unit_plus |
$ATRSPREC/bin/socket -q 127.0.0.1 24142")
```

Using the program `startclient.py`, `loginclient.py` forks three processes:

- `SBpara`, an ATRSPREC client program for speech recording, preprocessing and parameter compression,
- `chatrplayer.py` a program to read, convert and play CHATR synthesized speech and
- `MatrixGui.py`, the ATR-MATRIX graphical user interface.

These programs are described in Appendix B .

Perceptable events during startup are the playing of a welcome audio message and the GUI window being opened (and usually filling the whole screen). After that, the user can control the behaviour of ATR-MATRIX using the GUI buttons and commands. Figure 8 shows a screenshot of a Windows 98 desktop with the ATR-MATRIX client running, taken after the first utterance was recognized.

As explained in Section 2.2, the client programs have to control the audio device to avoid problems with half duplex machines. When the user clicks on the "Awake" button, the `ATRcontrol` module in `SBpara` gets a RPC telling it to start AD. The "Go to sleep" button causes AD to stop. Also, while `chatrplayer.py` is receiving or playing synthesized data from CHATR, it must be ensured that AD is turned off. For this purpose, `chatrplayer.py` keeps track of the current awake/sleep status and sends RPCs to `SBpara` if necessary. Figure 9 gives an example sequence of RPCs.

The client can exit the session by selecting "exit" in the file menu of the GUI. This closes down all components of all clients (including "listen-in" clients) connected to this server.

Figure 8: A screenshot of the ATR-MATRIX client running on a Windows 98 machine.

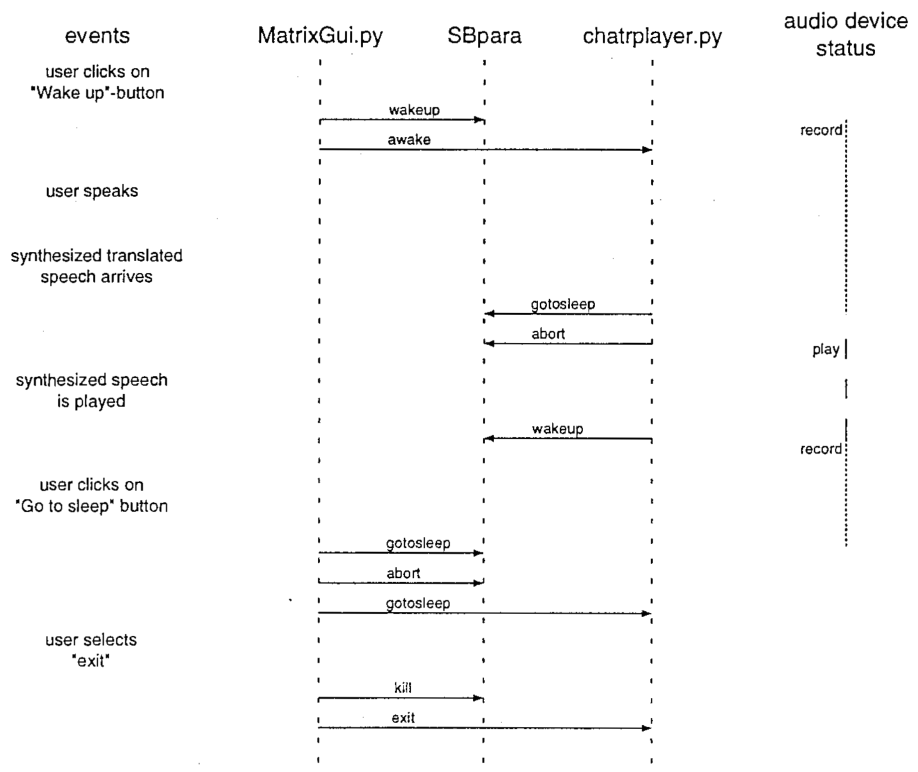

Figure 9: An example for a sequence of events and the resulting RPCs between the client components.

# 4 Telephone-Based ATR-MATRIX

We also implemented a rough prototype for a phone-based ATR-MATRIX client-server system. The system requires as client only a telephone. Acoustic models were trained by using downsampled and telephone-filtered data that had been collected with a high-quality microphone at 16 kHz[Woudenberg, 1994].

As interface to telephone line we used a Dialogic D/41ESC board on a PC with Windows NT. The hardware companie's interface libraries are in C/C++. The DLin ("Dialogic Interface") python library [Biem et al., 1999] allowed access to the C/C++ libraries from python. The phoneserver is written in python.

There are two ways a phone-based ATR-MATRIX system can be conceived:

- a one-user system, with the client sending speech and the server generating the synthesized translation.
- the ATR-MATRIX server can connect automatically via the communication switch to other clients, enabling a dialog between the phone client and other clients, using either a phone, wearable or normal computer. In this case, the server would synthesize translated answer from the dialog partners. Figure 10 explains the timeline of such a system.

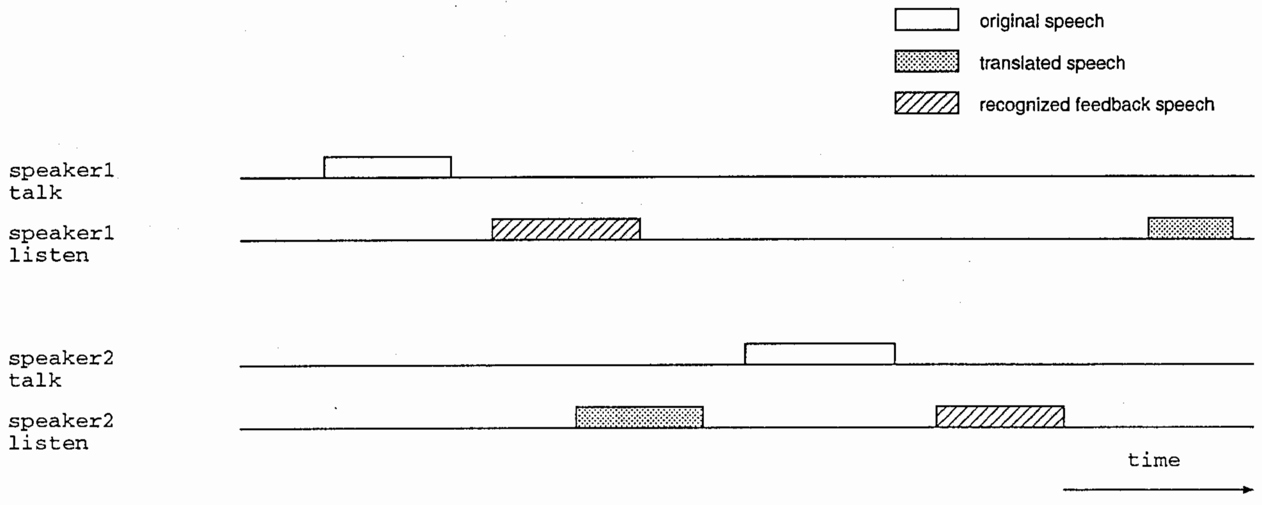

Figure 10: Timeline for one turn using recognition feedback for a display-less system

One important aspect of a speech translation system is user feedback. Especially, for a display-less system, information about status becomes important.

A possible approach is to send various kinds of tones or audio signals as "background noise" symbolizing the current status of the recognition system, e.g. to decide between "recognizing" and "waiting" state, but in the current prototype this is not implemented.

Figure 11 shows a schematic overview of a client-server system with a telephone client. The unused socket server on the server is for the graphical user interface information. The client can not make use of it, but a system administrator or audience at a demo can follow the conversation using it. The socket server is also used to initialize the ATR-MATRIX server, i.e. for sending a "wake up" signal.

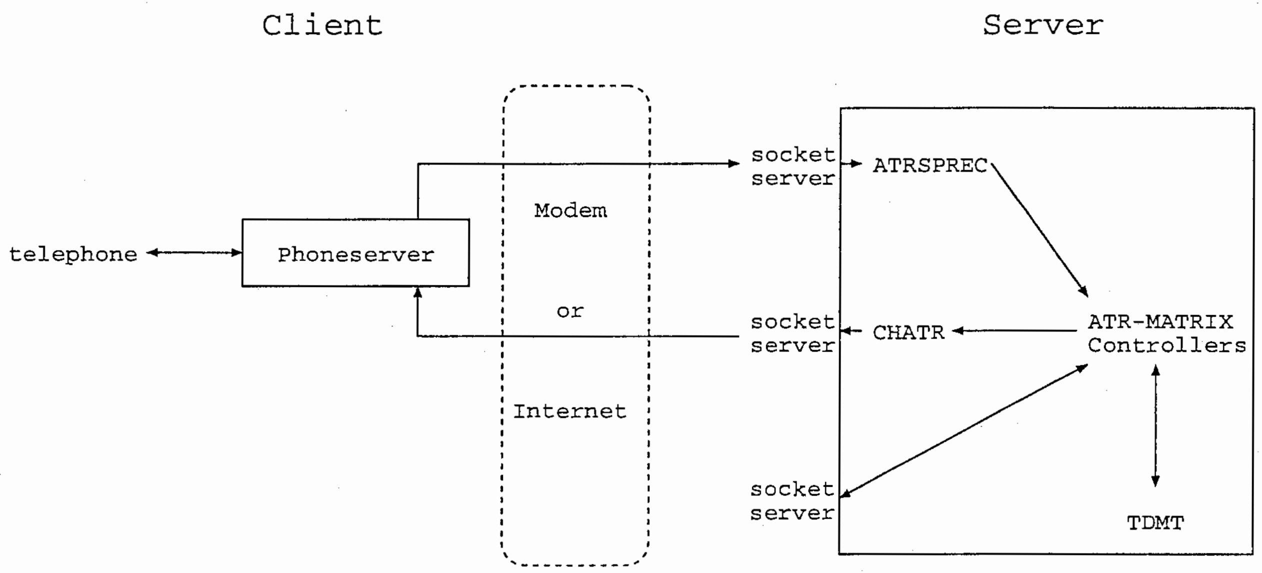

Figure 11: Overview over a telephone based client-server system

# 5 Evaluation

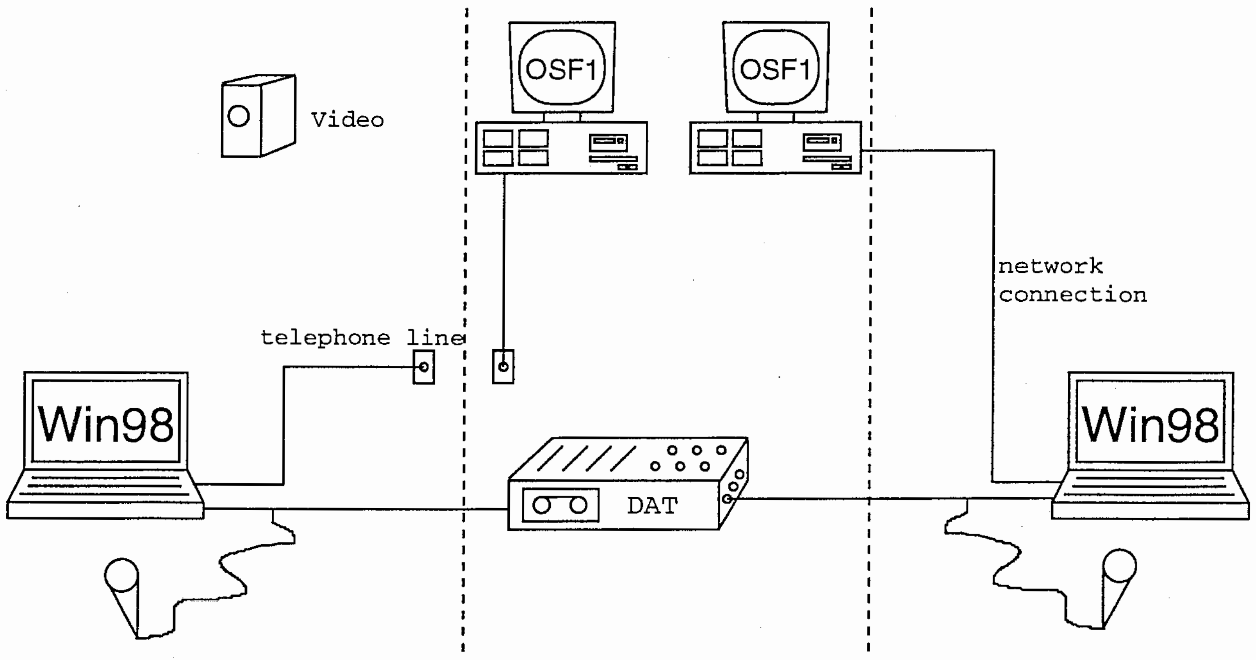

Figure 12: Overview of the client-server ATR-MATRIX evaluation experiment setup.

## 5.1 Experiment Setup

Evaluation experiments were performed to find out how easy the system is usable for an unexperienced user, hereforth called "customer" and which problems arise from unexpected use. The Japanese users were each asked to reserve a hotel room using ATR-MATRIX. A second ATR-MATRIX system was given to an experienced English speaker who acted as hotel clerk.

Each Japanese customer was given one page technical manual how to use ATR-MATRIX (see Appendix G.3) and some pages of dialog content information such as a schedule, telephone and credit card number. The customers had to start with a standalone laptop, i.e. they had to open a modem connection to the internet and to run ATR-MATRIX by themselves.

Figure 12 explains the experiment setup. Servers were a DEC alpha 500/500 with 500 MB RAM and Digital UNIX V4.0D for Japanese and a Compaq workstation XP-1000 with 400 MB RAM running Digital UNIX V4.0E for English. Client computers were a Panasonic CF-S22 Laptop with Intel MMX processor (266 MHz), 95 MB memory and built-in 56kBaud modem for Japanese and a Panasonic AL-N2 Laptop with Intel Pentium 133 and LAN network connection for the English side.

Software versions were sprecwin r06r04i on the clients and modified ATR-MATRIX 2.4.0 on the servers. Modifications compared to ATR-MATRIX 2.4.0 include update to ATRSPREC version r06r04f, TDMT release CstarII-9905_2, bugfixes in matrix and the changes necessary to turn ATR-MATRIX into a server as described in Appendix A.4.

Acoustic models were the gender dependend models as released on July 13th 1999 for Japanese and a speaker-adapted model for the English speaker. Language models were the release from June 14th 1999 for Japanese and the release from June 11th 1999 for English.

## 5.2 Results

In the experiments, the users had little problems with the setup of ATR-MATRIX. Still, during experiments, some difficulties appeared. Occured problems were:

- For one experiment, the telephone line to the ATR modem cascade was busy. This experiment was interrupted and restarted with a different telephone number.
- One customer started ATR-MATRIX four times because she was unsure if she had double-clicked the start icon properly or not. The resulting CPU load caused a crash of Windows. This problem could be prevented by e.g. a window that opens up directly after the icon was doubleclicked with contains

a message like "starting...". Alternatively the startup-process could be locking, i.e. prevent other startups at the same time.

- Customers tended to start ATR-MATRIX too early. The expected order is to start the modem connection, wait until dialling, modem negotiation and login are completed and then doubleclick the startmatrix icon. Some customers started ATR-MATRIX directly after the whisteling noise from the modem negotiation stopped. The laptop is not yet connected to the network at that time, so that socket connections cannot be opened. But the number of retries and the waiting time before retrying to connect proved to be sufficient to ensure safe startup.
- The screen saver on the client computer was not turned off correctly in the beginning.
- The most time-consuming part of the dialog were long numbers like credit card or telephone numbers. A problem was that customers tried to say the whole number at one time, and also repeated the whole number in case of misrecognition. Better would have been an improved grammar for them, or else a good way to enter them, for example "please tell me your visa card number, four digits at a time.". This is mentioned in the usage explanations but will need to be pointed out more in further experiments or demos. Appendix G.2 gives detailled information.
- The "Go to sleep"-button did not seem to have immediate effect and thus had only reduced applicability as cancel function.

Table 2 shows a list of customers and how long it took them to complete the task, i.e. to reserve a hotel room. Most customers finished the reservation conversation after some 40 minutes. The total conversation time splits into setup time, including connecting the computer via modem to the network and starting ATR-MATRIX, and the actual dialog time. Customer F0001 could not connect to the dial-in network because the modem line was busy, so the setup time for her is very long.

Table 2: Customers and their performance

| customer ID | sex | setup time | dialog time | turns | total time |
|---|---|---|---|---|---|
| M0001 | M | 3:07 | 35:11 | 40 | 38:18 |
| F0001 | F | 15:26 | 55:28 | 80 | 1:10:54 |
| F0002 | F | 2:06 | 38:19 | 79 | 40:25 |
| F0003 | F | 4:20 | 49:55 | 72 | 54:15 |

Table 3 shows recognition rates. They are computed using the recorded speech and manually generated transcriptions (see Appendix G ). As the speech was recorded on DAT before passing through the client audio device, the calculated accuracy does not match the real recognition results during the experiment.

Table 3: Recognition rates for speech as recorded on DAT tapes

| Customer | word accuracy | word correct | net accuracy | net correct |
|---|---|---|---|---|
| M0001 | 88.28 | 93.45 | 91.03 | 95.86 |
| F0001 | 85.32 | 91.86 | 87.21 | 93.60 |
| F0002 | 72.86 | 85.32 | 76.95 | 87.55 |
| F0003 | 44.03 | 53.67 | 53.46 | 62.26 |
| average | 72.50 | 81.18 | 76.92 | 84.80 |

To have more precise information about the actual recognition rate the speech that was recorded on DAT was used as input to the microphone plug of the laptop computer used in the experiment. Table 4 shows the recognition results aquired with this setting. The laptop had a OPL3-SA3 audio device.

To measure the difference the laptop audio device makes for recognition accuracy, the same speech data was filtered through one more laptop, itlnpc70. Table 5 shows the results.

## 5.3 Discussion

Usually, the main purpose of evaluation experiments is to get recognition rates, translation accuracy ratings etc. for untrained speakers. These figures can be calculated for this experiment, too, but they are statistically

Table 4: Recognition rates for speech filtered by the client laptop audio device

| Customer | word accuracy | word correct | net accuracy | net correct |
|---|---|---|---|---|
| M0001 | 88.62 | 92.41 | 91.72 | 95.17 |
| F0001 | 83.50 | 91.53 | 87.30 | 94.16 |
| F0002 | 77.51 | 85.69 | 83.27 | 89.59 |
| F0003 | 48.22 | 58.91 | 55.77 | 64.57 |
| average | 74.17 | 82.26 | 79.30 | 85.98 |

Table 5: Recognition rates for speech that was piped through a Gateway 5150 with a ESS1876 audio device.

| Customer | word accuracy | word correct | net accuracy | net correct |
|---|---|---|---|---|
| M0001 | 89.66 | 93.79 | 91.03 | 95.17 |
| F0001 | 83.80 | 91.24 | 86.72 | 93.58 |
| F0002 | 81.41 | 88.85 | 84.57 | 90.89 |
| F0003 | 47.38 | 58.49 | 56.39 | 66.25 |
| average | 75.28 | 83.12 | 79.50 | 86.53 |

irrelevant, as only four speakers were used. In previous ITL evaluations of ITL, the first few dialogs are not actually used to calculate recognition rates etc. as they are utilized to find out configuration and setup problems. Furthermore, the author's lack of experience with evaluation experiments caused additional difficulties.

The main objective of the evaluation experiment reported here was to find out problems occuring when people unfamiliar with ATR-MATRIX and the client software are asked to use it. For this purpose, the first experiments are the most valuable ones. The experiments provided insight in the process of evaluation experiments. Appendix G gives further details about which points have to be taken care of for preparation of evaluation experiments and which steps are necessary to retrieve the desired results.

All four speakers completed their task, although the task completion times were far too long. In the "real" world, they would have probably given up.

Interestingly, speaker F0003, with a recognition accuracy of under 50% took less time to complete the task than speaker F0001 who achieved more than 80% word accuracy. In other words, word accuracy and task completion rate are not necessarily highly correlated.

The influence of the audio device is amazingly small. The significant drop in recognition rate we expected to find if we use "laptop audio device speech" instead of high quality speech directly from DAT did not occur. On the contrary we got slight recognition rate improvements with speech piped through a laptop audio device.

Compared to a simple phone call, "setup time" is also too long, i.e. between 2 and 4 minutes if nothing goes wrong. This is partly because the customers were unsure of what to do and read the instructions carefully before each of the single steps.

In summary, there is still a lot of effort required for building a useful speech translation product.

## 6 Future Work

Major unsolved problems are:

**audio device quality:** in a client-server setting, control of the client audio device becomes very difficult. An informal test with 8 different laptops showed that the quality of the AD was quite bad for 6 of them, with audible differences even for the same model. We hope that USB based audio devices will solve this problem.

**status display and feedback:** the current status of the server should be fed-back to the user. In the standalone ATR-MATRIX system, this is achieved by an additional video channel and 100 msec updates of the current recognition status. This would require too much bandwidth in the client-server setting.

**evaluation:** overall system evaluation is a time-consuming process that requires bi-lingual speakers to manually grade the translation quality. Automatic evaluation for translation results is still an open research area.

**porting:** we consider to port the client side to other light-weight operating systems, like WindowsCE. Also, the use of telephone as input medium to the server needs more research efforts.

**multiple servers on one host:** for industrial use it is important to run several servers on one computer. Currently the use of fixed socket numbers and temporary files make it impossible to run more than one server on the same computer.

Furthermore, there are some structural disadvantages in the client-server structure that will no be solved without considerable research work:

- To save bandwidth, only cepstral data is sent from client to server, not speech. This makes it impossible to listen to the actually spoken utterance, thus prohibiting transcription of data collected with client-server. Pitch information for prosodic analysis is also unavailable. Sending extremely compressed speech along with the cepstral data may help with this problem.
- the status of the server is unknown. The information, which part of ATR-MATRIX is working at the very moment, or if the recognition of the last utterance has been aborted, is only available on the server. Clients have to cope with an uncertain feeling about the server state.
- To implement some features of client-server ATR-MATRIX, some cross-module accesses had to be implemented. This fixes ATR-MATRIX to the use of ATR-SPREC and CHATR. Originally, ATR-MATRIX was designed so that only the controller of one module was program-specific, making it easy to replace modules. Now several programs need to be modified.

## 7 Conclusion

In this report, we described the successful implementation of a client-server architecture. The client-server architecture allows many new features for ATR-MATRIX, including

- availability under Windows 95/98 additional to Linux and OSF1,
- reduced hardware requirements, e.g. a note PC with Pentium 133 processor,
- easy installation, start and shutdown,
- support for non-standard computers such as wearable or Windows CE computers or phone clients,
- applicable even on laptops with half-duplex sound devices,
- efficient use of servers because of resource management system,
- availability anywhere in the world if provided a (cellular) phone and a note PC.

## Acknowledgment

The authors would like to thank all members and the management of ATR ITL to support us in this applied research effort. In particular, we would like to thank the members of the "Technical Support Group" i.e. Benjamin Reaves, Atsushi Nishino, Koji Takashima and Takeshi Matsuda. We also extend our gratitude to Ryosuke Iwasawa for programming support and to Hirofumi Yamamoto for the design of the SPREC eventloop that made the splitting of the speech recognizer very easy.

## References

[Biem et al., 1999] Biem, A., McDermott, E., and Woudenberg, E. (1999). The ATR HIP minimum error classifier system (MECS). Technical report, ATR Human Information Processing Research Laboratories. (to be published).

[C-Star, 1999] C-Star (1999). Consortium for Speech Translation Advanced Research (C-Star): C-Star Experiment. http://www.c-star.org/.

[Gruhn and Singer, 1998] Gruhn, R. and Singer, H. (1998). Scalar quantization of cepstral parameters for low bandwidth client-server speech recognition systems. Technical Report TR-IT-0286, ATR.

[Gruhn et al., 1999] Gruhn, R., Singer, H., and Sagisaka, Y. (1999). Scalar quantization of cepstral parameters for low bandwidth client-server speech recognition systems. In *Proc. Acoust. Soc. Jap.*, pages 129–130.

[Handley and Jacobson, 1998] Handley, M. and Jacobson, V. (1998). SDP:session description protocol. Internet-draft RFC 2327, Internet Engineering Task Force. ftp://ftp.isi.edu/in-notes/rfc2327.txt.

[Handley et al., 1997] Handley, M., Schulzrinne, H., and Schooler, E. (1997). SIP:session initiation protocol. Internet-draft RFC 2543, Internet Engineering Task Force. ftp://ftp.isi.edu/confctrl/docs/draft-ietf-mmusic-sip-02.ps.

[Reaves et al., 1999] Reaves, B., Nishino, A., Singer, H., Fujisawa, K., Yamada, S., Sugaya, F., Takezawa, T., Yokoo, A., and Yamamoto, S. (1999). ATR-MATRIX: a speech translation system between English and Japanese. In *Proc. Info. Proc. Soc. Jap.*, pages 87–88, Tokyo.

[Reaves et al., 1998] Reaves, B., Nishino, A., and Takezawa, T. (1998). ATR-MATRIX: Implementation of a speech translation system. In *Proc. Acoust. Soc. Jap.*, pages 53–54.

[Stevens, 1994] Stevens, W. (1994). *TCP/IP Illustrated.* Addison–Wesley Publishers.

[Sugaya et al., 1999] Sugaya, F., Takezawa, T., Yokoo, A., and Yamamoto, S. (1999). Evaluation on bidirectional Japanese and English ATR-MATRIX. In *Proc. Acoust. Soc. Jap.*, pages 107–108. (in Japanese).

[Takezawa et al., 1998] Takezawa, T., T.Morimoto, Sagisaka, Y., Campbell, N., Iida, H., Sugaya, F., Yokoo, A., and Yamamoto, S. (1998). A Japanese-to-English speech translation system: ATR-MATRIX. In *Proc. ICSLP*, pages 957–960.

[Woudenberg, 1994] Woudenberg, E. (1994). Telephone band conversion of studio quality audio data. Technical Report TR-H-109, ATR.

[Yomiuri Shimbun, 1999] Yomiuri Shimbun (1999). ハロー音声翻訳. No. 16735.

# A Installation Manuals

## A.1 Minimal Requirements for Client

To install and run the ATR-MATRIX client, you need at least:

**CPU** Pentium processor with 133 MHz

**operating system** Linux, OSF1 or (Japanese) Windows 95/98

**disk space** about 20 MByte

**network** ethernet card or modem with at least 9600 Baud bandwidth.

## A.2 How to Install an ATR-MATRIX Client on a Windows Computer

### (A.2.1) Necessary Installation

The installation needs about 20MB (10 MB for python, 10MB for tcl/tk) and all files are in `c:\Program Files\Python` and `c:\Program Files\Tcl`.

From WINITL network, the easiest way is to execute the following files in the following order

```
\\nessie\share\dept1\singer\py152.exe            # install python
\\nessie\share\dept1\singer\py152tcl805jp.exe    # install some additional libraries
\\nessie\share\dept1\singer\sprecwinr06r04i.exe  # install selected SPREC components
rm c:\temp                                       # delete temporary directory
```

The installation will also create the icons `startmatrix` and `startmatrix English` on your desktop. The system is dependent on a certain path structure: just say yes to all questions without changing any setting.

### (A.2.2) Optional Installation

If you have enough disk space, you should also install the CHATR database so that output comes faster. Here is how you do it:

This batch files will install the necessary CHATR components for a ATR-MATRIX client-server system where only the information about units is transferred (instead of actual speech). This reduces the bit rate from 256 kbps (16kHz 16bit) or 64 kbps (ulaw) to 1 kbps.

If you only need to do Japanese to English translation, you only need the 2 English speakers nes (male) and mlp (female), these two databases total 197 MB. To install them, run

```
\\nessie\share\dept1\work1\singer\chatr2e.bat
```

If you want to allow both Japanese and English synthesis, you can install 2 Japanese speakers (takes about additional 550 MByte):

```
\\nessie\share\dept1\work1\singer\chatr2e2j.bat
```

A different way to have CHATR on the client is to use a CD with the CHATR database. This saves disk space, for the price of being a little slower due to the startup time of the CD drive. As CHATR must be in the path `c:\chatr`, some setup must be done to enable CD use. The steps are:

- insert the CD in the CD drive
- doubleclick on chatrsetup in the directory `c:\program files\python\sprecwin\ATRmatrix`.
- a DOS window pops up, asking where the CHATR database is that you want to use. Type in the full path, e.g. if your CD drive has the letter `R:`, type `r:\chatr`.
- the installation is done, test it.

### (A.2.3) How to Start an ATR-MATRIX Client

There are two ways to cleanly startup a client:

- doubleclick on `c:\program files\python\sprecwin\ATRmatrix\loginclient.py`
- doubleclick on the `startmatrix` icon on your desktop.

### (A.2.4) Known Problems

Very often, the recording levels under Windows are set to mute by default. If the ATR-MATRIX client is used for the first time, this may cause confusion. The user must make sure that audio in and out is enabled and the volume at an appropriate level.

## A.3 How to Install an ATR-MATRIX Client on a UNIX Computer

### (A.3.1) Install ATRSPREC

First, install ATRSPREC. Inside the ATR network, the source can be retrieved with the command

```
cvs -d atra22:/home/singer/CVSROOT checkout SPREC
```

Then follow the instructions in the file `SPREC/README`. In addition, start `make` in the following directories:

```
$ATRSPREC/etc/ADDA
$ATRSPREC/etc/VC++/SBinput
```

### (A.3.2) Install Other Software

Make sure the following programs are available and in the search path:

- `sox` (audio file format conversion)
- `untoast` (for GSM decompression, see Appendix F )
- `concat` (for chatr wave concatenation, get it from dept. 2 of ITL)
- (Linux only) `adda`
- (OSF1 only) `audioplay` as set in the environment variable `SPREC_DAOUT` (source `$ATRSPREC/etc/Cshrc` if this variable is not set).
- (OSF1 only) the record program as set in the environment variable `SPREC_ADIN`.

### (A.3.3) How to Start an ATR-MATRIX Client using csh

```
source $ATRSPREC/etc/Cshrc
setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH":.
cd SPREC/sample/ATRmatrix
loginclient.py
```

### (A.3.4) Special Cases

Some clients need special preparations before list of start commands work.

**start client from remote computer using rsh** In this case, the following comands are necessary to be able to access a local datlink. This example assumes `atra57` to be the computer `loginclient.py` will be started on. `pcx196` is the computer the user is physically located at and where the datlink is connected to.

```
rlogin atra57
setenv AUDIO_DEVICE pcx196:0
setenv DISPLAY pcx196:0
/usr/local/datlink/bin/naserver & # if not running already
```

One known problem with an OSF1 machine as remote computer is that all CHATR speech will be played on the remote computer (e.g.atra57) as `audioplay` plays locally. Set the environment variable `SPREC_DAOUT` to the play command on your computer, started with `rsh`.

**machine without mounts to other ATR disks** Normally, if a UNIX client uses `concat` to generate waves from chatr units, the wavefile library is accessed over the ATR network. If this is not possible, you have to copy it and modify some configuration files. Assuming the chatr database to be copied to `/tmp/mychatrdbs`, a set of commands is:

```
setenv SPEAKERTOP /tmp/mychatrdbs/
cp -r /home/as58/srinivas/patrick/chatr/lib/dbs/ $SPEAKERTOP/
vi $SPEAKERTOP/nes/nes.concat
vi $SPEAKERTOP/fmp/fmp.concat
vi $SPEAKERTOP/mlp/mlp.concat
vi $SPEAKERTOP/mys/mys.concat
```

In the `*.concat` files, you have to replace the line with the wavefile library path, e.g.

```
/homes/srinivas/patrick/chatr/lib/dbs/nes/nes.wavlib
```

by your path (use of environment variables such as $SPEAKERTOP is not supported), e.g.

```
/tmp/mychatrdbs/nes/nes.wavlib
```

Remember to set the `SPEAKERTOP` variable each time before starting the client.

### (A.3.5) Known Problems

On shutdown, some sockets do not exit properly. Use

```
ps auxww | grep socket
```

to find out their process IDs and kill them.

## A.4 How to Install an ATR-MATRIX Server on a UNIX Computer

This chapter is a step-by-step description how to install an ATR-MATRIX server.

- Installation: The first step is to install ATR-MATRIX using the most current installation script as provided by TSG.

  Then test if it works.

- Config files: Then, modify three configuration files.

  1. The ATRSPREC config file must be changed so that the server part of SPREC expects compressed cepstral data as input.

     ```
     vi $MATRIX_PREFIX/run/config.j
     ```

     replace the section I/Ocontrol for example by:

     ```
     I/Ocontrol:inputFormat=FrameSync
     I/Ocontrol:inputParamSize=13
     I/Ocontrol:inputParamType=float
     I/Ocontrol:inputFd=process("$ATRSPREC/sample/ATRcollect/atrCollectDoubleSocket.py \
                                 -portr=24042 -stdout=OK -portws=24442")
     I/Ocontrol:inputEOFexit=OFF
     I/Ocontrol:inputByteorder=BigEndian
     I/Ocontrol:outputFormat=NULL
     I/Ocontrol:outputFd=stdout
     I/Ocontrol:outputByteorder=BigEndian
     I/Ocontrol:rpcNumber=5
     I/Ocontrol:inputDecompress=ON
     I/Ocontrol:outputCompress=OFF
     I/Ocontrol:inputDecodeBookFile=$ATRSPREC/etc/VC++/SBinput/codebook.cms56dec
     I/Ocontrol:outputEncodeBookFile=$ATRSPREC/etc/VC++/SBinput/codebook.cms56enc
     ```

     Also make sure that the used models are trained with CMS. You may also need to copy the codebook `codebook.cms56dec`, e.g. from `/home/rgruhn/ATRCOMPRESS/codebooks/codebook.cms56dec`.

  2. Next, modify the CHATR configuration file, so that the output is sent via a socket to the doubleserver:

     ```
     vi $MATRIX_PREFIX/run/ChatrConfigFile.cstarII
     ```

The following line should be added:

```
(Audio Command "cat $FILE | $ATRSPREC/bin/socket -q localhost 24142")
```

3. The last config file to modify is the SipConfigFile:

   ```
   vi $MATRIX_PREFIX/run/SipConfigFile
   ```

   Instead of `MatrixGui.py`, we want `GUIserversocket.py` to be started by the `GUIcontroller`, so we change

   ```
   SipGuiController.GuiProgram : MatrixGui.py
   ```

   to

   ```
   SipGuiController.GuiProgram : GUIserversocket.py
   ```

- Copies: copy the files `doubleserver.py`, `chattr.py` and `GUIserversocket.py` so that they are in the search path and in an appropriate place, e.g.:

```
cp $MATRIX_PREFIX/resources/SPREC/sample/ATRmatrix/GUIserversocket.py \
   $MATRIX_PREFIX/resources/SPREC/sample/ATRmatrix/doubleserver.py \
   $MATRIX_PREFIX/resources/matrix/python/

cp $MATRIX_PREFIX/resources/SPREC/sample/ATRmatrix/chattr.py \
   $MATRIX_PREFIX/resources/SPREC/scripts/python/lib/
```

- Change ATRSPREC executable: We want an ATRSPREC that starts with the `cep2para` module, so we have to change a makefile and recompile.

```
vi $MATRIX_PREFIX/resources/sprec97/src/sprec/makefile
```

In this file, change the line

```
Modules="ATRepd,ATRlattice,sprec97,ATRwave2cep,ATRcep2para,ATRdisplaypow"
```

to

```
Modules="ATRcep2para,ATRlattice,sprec97"
```

(order is important !) and recompile

```
cd $MATRIX_PREFIX/resources/sprec97-990312-MATRIX_2_3_2a/src/
make NOSPREC=1 STATIC_LINK=1 GOFLAG=-g
```

Note: if you are using matrix2.3.2 or older, make sure that in this makefile under `Libs=\` the library `-lATRcompress` is written after `-lATRevent`, not before.

- only matrix 2.3.2: change CHATR Controller (bug has been fixed in matrix 2.4.0 and higher) To enable Chatr to output the prosodic units (additional to the wave file), the CHATR controller has to be modified:

```
cd $MATRIX_PREFIX/resources/matrix/library/libsip++/src
vi SipChatrModule.cc
```

replace all patterns like

```
        sayString  = "(Say (Synth (Utterance Text \"";
        sayString += string;
        sayString += "\")))\n";
```

by

```
        sayString  = "(Synth (Utterance Text \"";
        sayString += string;
        sayString += "\"))(Save UnitLabels+ '/tmp/chatrjunk)(Say)\n";
```

(appears 7 times with different utterance units, Text, Phoneme, PhonoForm)

- Test: A possible set of start commands for a matrix server is:

```
cd $MATRIX_PREFIX/run
bash
source environment.bash
doubleserver.py >& /tmp/doubleserver.log &
SipMainController -config SipConfigFile >& /tmp/matrixserver.log &
```

- Other tricks:
  - when testing, mind those undead chatr and sockets and atrCollectDoubleSocket.py. After killing the server, sometimes some sockets or chatr keep running. Check the running processes using `ps` and the socket status with `netstat | head 15`.
  - on the server, the programs `sox`, `toast` must be available, on a Unix client `untoast` (and for OSF1 `audioplay`).
  - to actually use the server, you have to make an entry in the resource manager `matrixlogon.py` (see Appendix B.2).
  - for recompiling, remember to set the variables

```
export SIP_RESOURCE=$MATRIX_RESOURCES
export GFILTER=$GFILTER_LIB/..
```

# B Software Manuals

This chapter contains descriptions of the programs for client-server. Figure 13 shows a overview, which programs are used on client and server side.

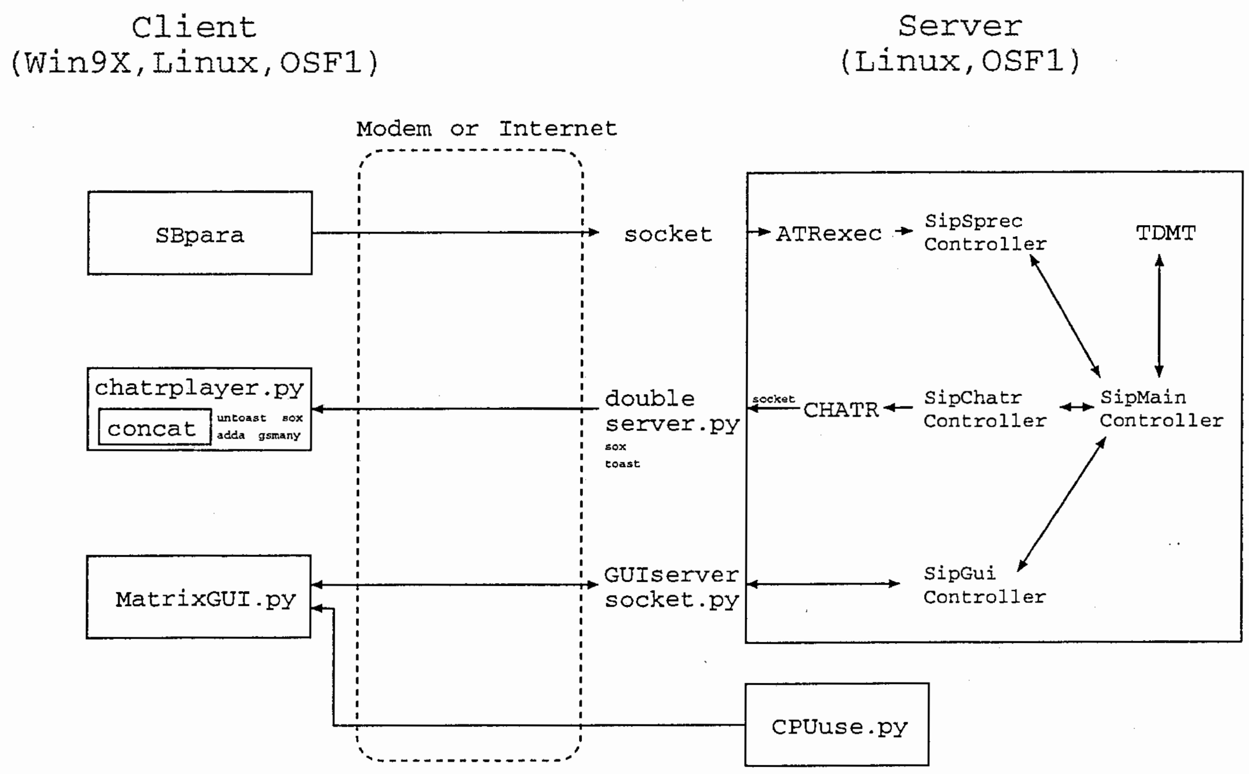

Figure 13: Overview over the used programs on client and server.

## B.1 loginclient.py

This is the program started if the `startmatrix` icon on the Windows desktop is doubleclicked. The installation procedure is described in Appendix A . It opens up a connection to the resource manager `matrixlogon.py`, typically running on `pcx196.itl.atr.co.jp:42000`, transmits the client properties and receives the IP address of a server. Before starting the actual ATR-MATRIX client programs, it configures CHATR with regard to whether the CHATR database is available on the client or not.

Table 6 lists the command line options of `loginclient.py`.

Usage example (Windows):

```
cd "c:\program files\python\sprecwin\atrmatrix"
python loginclient.py -inputlanguage=e
```

Usage example (Linux):

```
cd $ATRSPREC/sample/ATRmatrix
loginclient.py -inputlanguage=e -verbose_mode=off
```

## B.2 matrixlogon.py

`matrixlogon.py` is a resource manager that controls one or more matrix servers. It waits for login request on the "to all clients well known port" `pcx196.itl.atr.co.jp:42000` (default value). Login requests are made in SIP-type message format. Messages contain information about client capabilities such as whether chatr is installed as well as which service is desired, Japanese-English translation or vice versa.

Table 6: Configuration of `loginclient.py`

| option | default | description |
|---|---|---|
| `-config` | (none) | name of file with config options. |
| `-loginhost` | 133.186.35.196 | host to login to the resource manager. |
| `-loginport` | 42000 | port to login to the resource manager. |
| `-logoutport` | (none) | port where loginclient will wait for an exit signal from `MatrixGui.py`. If not set, it is chosen by random. |
| `-guiserverport` | 50002 | port where `GUIserversocket.py` is waiting on server machine. |
| `-doubleserverport` | 5598 | port where `doubleserver.py` is waiting on server machine. |
| `-messageversion` | 2 | version of login message (for future use). |
| `-matrixversion` | 2.4 | version software on server side the client expects (for future use). |
| `-inputlanguage` | j | language the client speaks, `j` or `e`. |
| `-clienttype` | computer | communication medium between client and server, `computer` or `phone`. |
| `-preferredhost` | (none) | if the client wants to use a specific host, he can set it using this option. If the server is busy or unknown, matrixlogon.py will return a different host fulfilling the requirements. |
| `-verbose_mode` | on | print additional messages to `stderr`. |

## (B.2.1) RPCs

Using RPCs, matrixlogon.py can be modified while running. The default RPC port is 5599 (which is ATRSPREC RPC-number 44).

RPCs are usually generated using `$ATRSPREC/bin/ATRsend_RPC`. The command line is structured like:

```
$ATRSPREC/bin/ATRsend_RPC -host=pcx196.itl.atr.co.jp
                          -rpcNumber=44 -receiver=anyone
                          -command="command commandoptions"
```

The following RPC commands are supported:

**exit** This RPC kills `matrixlogon.py`. All connected clients get the message `500 internal server error`, then all opened sockets are closed. `kill` and `BYE` commands have the same effect. Usage example:

```
$ATRSPREC/bin/ATRsend_RPC -host=pcx196 -rpcNumber=44 -receiver=anyone -command=kill
```

**status** Status prints out the list of known servers and their properties to stderr on server. As optional parameter, a host and port can be specified, where the status is also written to. Usage examples:

```
$ATRSPREC/bin/ATRsend_RPC -host=pcx196 -rpcNumber=44 -receiver=anyone -command=status
```

or

```
$ATRSPREC/bin/socket -sl 12345 &
$ATRSPREC/bin/ATRsend_RPC -host=pcx196 -rpcNumber=44 -receiver=anyone
                          -command="status pcx200:12345"
```

The output will look like the following example:

```
pau04.itl.atr.co.jp : {'clienttype': 'computer', 'socket': <atrspawn._Socket instance at 810b8c0:
                      'usedby': '133.186.32.202', 'lang': 'j'}
pau06.itl.atr.co.jp : {'clienttype': 'computer', 'socket': '', 'usedby': '', 'lang': 'e'}
```

The server database is sorted by server IP names. For each server, there is a python-style dictionary with the server attributes, including:

- `clienttype`, (whether the server expects a telephone or a computer as medium to the client)
- `usedby` (IP address of client who uses this server, empty string means that the server is unused)
- `lang` (expected input language to recognizer)
- `socket` (socket connection to client, this does not contain much information for the user but is included for completeness reasons)

**ping** This command has the format `ping host:port`. It is used to check if matrixlogon.py has crashed. Ping sends a pong, containing the string `pong` and a timestamp, to the IP address and port specified. Usage example:

```
$ATRSPREC/bin/ATRsend_RPC -host=pcx196 -rpcNumber=44 -receiver=anyone
                          -command="ping 133.186.30.219:12345"
```

**add** The add command adds a server to the list of known servers. The command has the format `add IPname:clienttype:inputlanguage`, with `clienttype` being one of (`computer`,`phone`) and `inputlanguage` one of (`e`,`j`). If an already known IP name is given, the existing setting is overwritten by the new one and the server marked as unused; this can be used as a reset. As Windows 95 does not support IP addresses (in number form), servers should be identified with their DNA names. Usage example:

```
$ATRSPREC/bin/ATRsend_RPC -host=pcx196 -rpcNumber=44 -receiver=anyone
                          -command="add atra57.itl.atr.co.jp:computer:j"
```

**remove** Remove a server from list of servers. The command has the format `remove IPname`. Note: This does not end possibly existing sessions between a client and the server. Usage example:

```
$ATRSPREC/bin/ATRsend_RPC -host=pcx196 -rpcNumber=44 -receiver=anyone
                          -command="remove itlnpc72.itl.atr.co.jp"
```

**kickclient** Force a client to exit. The command has the format `kickclient IPaddress`, with `IPaddress` being IP name or address of the client to kick out. This command searches the `usedby` field of all known servers for `IPaddress`. If the addresses match, the string `exit\n` is sent to the GUIserversocket on this server, causing all connected clients on this server to exit. Note: this only works for a normal session, it cannot fix situations in which server or client have crashed. Also, if the client does not actually exit, this RPC does not set the concerning server to unused; to enforce this, use the `add` RPC. It is recommended to use the `status` RPC to check the results of a kickclient call. Usage example:

```
$ATRSPREC/bin/ATRsend_RPC -host=pcx196 -rpcNumber=44 -receiver=anyone
                          -command="kickclient 133.186.30.201"
```

### (B.2.2) Configuration

The configuration can be done using a config file or command line parameters.

Table 7 lists the command line options of `matrixlogon.py`.

Table 7: Configuration of `matrixlogon.py`

| option | default | description |
|---|---|---|
| `-help` | (n.a.) | Display help message. |
| `-config` | (none) | name of file with config options. |
| `-loginport` | 42000 | port where login requests will be expected. It is recommended to leave the default as it is because all clients must know this port. |
| `-servers` | pau06.itl.atr.co.jp:computer:e | list of matrix server IP names and the server's settings, with a ":" separating between IP name and setting fields and a "," separating between the data for each server. |
| `-controlport` | 5599 | RPC port. |
| `-verbose_mode` | on | Print additional messages to `stderr`. |

Usage example:

```
$ATRSPREC/sample/ATRmatrix/matrixlogon.py -controlport=12345 -verbose_mode=off
```

## B.3 SBpara

SBpara is the client-side part of ATR-SPREC. It consists of the SPREC modules ATRepd (endpoint detection) and ATRwave2cep (preprocessing from speech signal to cepstral parameters). Additionally, SBinput (for audio input) and ATRcontrol (for administration) were programmed and included. Under Windows, SBpara directly accesses the sound device for AD, under other operating systems the program specified in the command line will be used.

Table 8 lists the command line options for SBpara, that were added to .

Table 8: Additional configuration file items of SBpara added for client-server

| option | description |
|---|---|
| `config` | name of a config file to read the configuration from. |
| `SBinput:SamplingFrequency` | input sampling frequency. |
| `SBinput:inputParamSize` | input parameter size. |
| `SBinput:outputDataType` | one of `ATRDwave`, `ATRDrwave` and `ATRDmwave`. |
| `SBinput:timeout` | timeout in 10ms frames after which the program terminates. Time is taken after record begins. A setting of -1 turns the timer off. |
| `SBinput:rsize` | size in 10ms frames of the ringbuffer storing input data if the server does not read it fast enough. |
| `SBinput:timecut` | |
| `ATRcontrol:autoStartRecord` | flag to automatically start recording on program execution. |
| `ATRcontrol:immediateCommand` | |
| `processAD` | process to use for audio recording (for Unix systems). |

Usage example:

```
SBpara -config=config.sbp -processAD="narecord -o left -p 16 -e linear -f raw -s 16000"
```

# C Performance Enhancements

The ATR-MATRIX client runs on any Computer with MS Windows, Linux or OSF1 operating system. However, this does not mean every setting is optimal just as it is in the standard installation. Recognition performance can be enhanced by tuning the following ATRSPREC client properties:

**cepstral mean subtraction (CMS)** The purpose of CMS is to equalize microphone and audio device properties. The default CMS file was made for the Sony ECM-MS907 condenser microphone on a Gateway Solo 5000 series laptop. It works well with the same microphone type on other laptops. Gain level setting also influences the effect of CMS.

**end point detection (EPD)** EPD means to search for the end and beginning of an utterance. It is depending on noise level both due to background noise and due to audio device properties. Wrong EPD settings cause speech to be cut off or extra words to be recognized at the end of utterances.

**silence model** The silence model in the acoustic model is strongly depending on the background noise. Its adaptation can improve the recognition rate and prevent background noise from being misrecognized as speech.

## C.1 Check EPD Settings on the Client

The following text is a manual which programs to start to check the EPD settings. For details about the settings and which modifications cause which effect read the EPD manual[3].

This section will describe the case of a Windows client. Users with a Linux or OSF1 client please read the explanations in `$ATRSPREC/etc/ATRepd.test/atrEpdTracker.py`.

For Windows clients, the check is spread over two computers becasue of display reasons (gnuplot). You need a Linux or OSF1 computer additionally to your Windows system. On the Unix system, start

```
bash
source $ATRSPREC/etc/Bashrc
$ATRSPREC/bin/ATRpython $ATRSPREC/etc/ATRepd.test/atrEpdTracker.py \
-I/Ocontrol:inputFd="socket(50000)" -config=config.mine
```

please note that `config.mine` must be a config file containing the same settings as on the client and additionally the lines:

```
Demo:adin="adda 8000 16 w"
Demo:daout="adda 8000 16 r"
Demo:inputByteorder=$SPREC_INPUTBYTEORDER
Demo:infileByteorder=BigEndian
Demo:machineByteorder=$SPREC_MACHINEBYTEORDER
Demo:outputByteorder=$SPREC_OUTPUTBYTEORDER
Demo:outfileByteorder=BigEndian
```

The `adin`- and `daout`-settings may vary depending on your application.

On the Windows client, please start

```
cd \progra~1\python\sprecwin\win
SBmain -config=config.sb -outputFd=socket(itlnpc70.itl.atr.co.jp:50000)
```

The server computer name `itlnpc70` must be changed to your server DNS name.

For demonstrations it has proved to be very helpful to create an icon to start the software easily.

## C.2 Silence Model Adaptation

For silence adaptation, install the software archive

```
\\nessie\share\dept1\singer\onlineSilence.zip
```

which also provides a file `README.txt` with instructions. A UNIX server is required additionally to the Windows client.

The basic steps are:

[3]http://www.itl.atr.co.jp/ singer/software/SPRECDOC/alpha/doc/tr/node30.html

- read the README.txt
- copy the program `online silence` to the server and start it.
- modify the batch file `SBParaX.bat` and start it.

# D FAQ and Troubleshooting

This chapter lists known problems and how to solve them.

## D.1 Problems Appearing on Client Side

- Q: I installed the ATR-MATRIX client as I should, I can start and see the GUI, but nothing's happening when I speak.

  A: Many computers have their sound devices set to mute by default. If you are on a Windows PC, check the sound control panel. Set the microphone to high gain level. Under Linux, please check your sound driver.

- Q: Under Windows, I doubleclicked on the startmatrix icon, and suddenly 8 new minimized windows appear in the task bar. They are flashing and/or Windows crashes.

  A: You clicked four times on the startmatrix icon instead of twice. This causes more than one client to be started. Windows cannot deal with this and hangs up.

  If you want to run more than one ATR-MATRIX client on the same computer, start them one after the other.

- Q: I must speak very loud to get the system recognize what I say. What can I do about it ?

  A: First, check the audio settings and raise mikrophone input to maximum. If it is already at maximum, you have to change the settings for the end point detection (EPD). You can find the current settings in the file `config.sbp`, which is in a standard Windows client installation in the path:

  ```
  c:\progra~1\python\sprecwin\win\config.sbp
  ```

  If the input gain is too low, a recommended measure is to set `lowerDispersionThreshold` and `energyThreshold` lower. Please consult the ATR-SPREC manual.

  The settings can also be changed (and the result be viewed with `atrOnlineDebug.py`) in a running session by sending RPCs to SBpara on the client. This is useful for debug, but all changes must be written into the config file, otherwise they are lost in the next session. An example for such an RPC is:

  ```
  $ATRSPREC/bin/ATRsend_RPC -host=133.186.32.202 -rpcNumber=7861 -receiver=ATRepd \
                            -command="OPTIONS lowerDispersionThreshold=10"
  ```

  Please mind that the `rpcNumber` is chosen by random. `loginclient.py` prints them to `stderr` on startup time.

- Q: My client does not have a large display, so I want a second client to show the GUI to audience in a demo. I also want the chatr speech.

  A: You can use `startclient.py` to do this. It starts only GUI and chatrplayer, but not SBpara. The commands are (for example):

  ```
  cd $ATRSPREC/sample/ATRmatrix
  startclient.py 0 pau04.itl.atr.co.jp
  ```

  with `pau04.itl.atr.co.jp` standing for the ATR-MATRIX server your client is connected to. The `0` is an option that prevents AD from being started.

## D.2 Problems Appearing on Server Side

- Q: The server seems to be connected to a client even after the client has logged off. Both matrixlogon and the ATR-MATRIX server still hold connections.

  A: There are two possibilities:

  - The client has logged off properly, but some programs did not exit. This can happen with Linux clients. To check if this has happened, log in at the clients computer and type

    ```
    netstat | head 15
    ```

If there is a connection to the server with the TCP/IP state ESTABLISHED, you know that there is a process left unkilled. Such a status report can look like the following example:

```
Proto Recv-Q Send-Q  Local Address           Foreign Address         (state)
tcp        0      0  pau06.24042             pcx200.19863            ESTABLISHED
```

Find out the leftover process with

```
ps -auxww | grep 24042
```

– The other possibility is that a client has logged off by simply switching off or cutting his modem connection without cleanly exiting ATR-MATRIX before. This is a TCP/IP problem. At the server computer, type

  ```
  netstat | head 15
  ```

  You will probably find lines in the state ESTABLISHED or FIN_WAIT_2.

  ```
  Proto Recv-Q Send-Q  Local Address           Foreign Address         (state)
  tcp        0      0  pau06.24042             pcx200.19863            ESTABLISHED
  tcp        0      0  pau06.50002             pcx200.19864            FIN_WAIT_2
  ```

  Sockets do not recover from these states, they wait forever, even if the other side switches off its computer. There are two ways to escape this connection state: one is, reboot the server machine (not only the software, the computer!). The other is to kill the ATR-MATRIX server, then once try to open up a TCP/IP connection (e.g. a socket or telnet connection) from the computer the client used to every affected port on the server (24042, 24242, 50002 are likely to be affected). This will change the socket state from FIN_WAIT_2 to TIME_WAIT. After about 1 minute waiting you can restart the ATR-MATRIX server.
  Details about TCP/IP connection states can be found in [Stevens, 1994]

- Q: What was the phone number of the phone client again ?

  A: 2018. Works only inside ATR.

- Q: How do I restart the telephone server `MatrixPhone.py` ?

  A: Do the following steps:

  1. kill the program with `control-c`. If it does not respond:
     (a) kill the window
     (b) open new `cygwin32` shell
     (c) `cd //f/src/dlin/harry`
  2. start `MatrixPhone.py`

- Q: I want to debug, but the server is so slow.

  A: If you are an expert user, you can speed it up by changing the SPREC configuration through a RPC:

  ```
  setenv MATRIX_RESOURCES /data3/atra51/itlusers/rgruhn/resources
  $ATRSPREC/bin/ATRsend_RPC -host=pau05 -rpcNumber=5 -receiver=ATRlattice\
   -command="OPTIONS beam=75,75 ATRlattice:lmscale=7,10 ATRlattice:amname=\
  $MATRIX_RESOURCES/sprec_j_model/amodel/19990202/MaleHMM/HMnet_filled.emb\
  .3.1500.bin,$MATRIX_RESOURCES/sprec_j_model/amodel/19990202/FemaleHMM/HM\
  net_filled.emb.3.1500.bin optionslist=stderr"
  ```

  Warning: If this RPC is wrong, e.g. one of the files does not exist, the server crashes.

  To check the current settings of lattice, type:

  ```
  $ATRSPREC/bin/ATRsend_RPC -host=pau07 -rpcNumber=5 -receiver=ATRlattice -command="STATUS"
  ```

- Q: I want to change the sprec models without rebooting the server.

  A: First, copy the new models to the local disk. Then, send an RPC to ATRlattice, for example:

```
$ATRSPREC/bin/ATRsend_RPC -host=pau07 -rpcNumber=5 -receiver=ATRlattice -command=\
"OPTIONS lexicon=$MATRIX_RESOURCES/sprec_e_model/lmodel/19990406/hrt.1000.lex.v3 \
ATRlattice:ngram=Class-2,$MATRIX_RESOURCES/sprec_e_model/lmodel/19990406/hrt.1000.\
bin.v3 optionslist=stderr"
```

Please make sure that the path is correct, as an incorrect setting crashes the server.

- Q: I send GUI commands to the server (port 50002) using a socket connection instead of GUI. Suddenly, ATRSPREC gets no input any more.

  A: Some commands (`connect`, `disconnect`) send a `gotosleep` command to ATRSPREC. Just also send an `awake` command to the server before speaking.

- Q: I want to use client-server in a local network, without connection to the ATR network. What do I have to take care of ?

  A: The resource manager. If you cannot access the ATR network, the login procedure will not work. The solution is to use your own `matrixlogon.py`. Add your local server IP names to the list of known servers and remove the others as desribed in Appendix B.2. A typical command line would be:

  ```
  matrixlogon.py -servers=mymatrixserver.itl.atr.co.jp:computer:e
  ```

  Also modify the start command for the clients:

  ```
  loginclient.py -loginhost=myloginserver.itl.atr.co.jp
  ```

- Q: If a client wants to open a connection to the Cstar communication server, he doesn't get the settings as they are written in the ATR-MATRIX server configuration file.

  A: Just reboot the server. Every time a client connects the GUI gets an initialization message. This does not come from the server itself, it is stored in the socket interface between GUI controller and GUI. This is a hack we pay for here: The initialization message is generated in two different parts of the GUI controller. Usually, both parts arrive immediately, but sometimes the second part comes too late and is not recognized as part of the init message. This second part mainly consists of the default settings for connections. To really fix this problem, rewrite the GUI controller(s).

- Q: How can I see the current server ATRSPREC configuration, e.g. of ATRlattice ?

  A: With an RPC to the target module, e.g.:

  ```
  $ATRSPREC/bin/ATRsend_RPC -host=pau07 -rpcNumber=5 -receiver=ATRlattice -command="STATUS"
  ```

- Q: I want to debug and need to block one or several servers for a short moment. How can I influence `matrixlogon.py` without deleting and adding to the server list ?

  A: It is cleaner to use the `remove` and `add` RPCs of `matrixlogon.py`. But if you really need something else, use a socket to send a login message to `matrixlogon.py`. This will mark one server as used (one server out of those matching your request). To free the server, kill the socket. An example is:

  ```
  echo "INVITE\r\ncontent-length:12\r\n\r\na=input:j\r\n" | \
       $ATRSPREC/bin/socket pcx196 42000
  ```

- Q: I want to check some (e.g. ATRSPREC) server settings. To be able to compare the results, I need identical input to ATRSPREC.

  A: First, start an ATR-MATRIX client under Linux. Make sure that displaypow sockets is not used.

  ```
  loginclient.py -inputlanguage=e
  ```

  Then record a compressed cepstral data file:

  ```
  $ATRSPREC/bin/socket -q pau06 24442 | tee yyy.cep | \
  $ATRSPREC/bin/ATRpython atrOnlineDebug.py -config=config.cepcmp -Modules=Cep2ParaModules
  ```

Kill the SBpara process. Then modify the server in the way you need, e.g. change ATRSPREC configuration with an RPC:

```
$ATRSPREC/bin/ATRsend_RPC -host=pau06 -rpcNumber=5 -receiver=ATRlattice \
      -command="OPTIONS beam=70,70 lmscale=7,10 active_model=1,2 \
      UTT_END_delay=55 word_boundary_skip=2 optionslist=stderr"
```

To test, send the generated cepstral data file to the server ATRSPREC:

```
$ATRSPREC/bin/socket -q pau06 24042 < yyy.cep
```

- Q: I have a file with compressed cepstral data that I read from the server on port 24442. How do I decompress that ?

  A: You can use ATRNULLexec to decompress it:

```
$ATRSPREC/bin/ATRNULLexec \
-config=/home/rgruhn/SPREC/sample/ATRNULLexec/configcomprfsync2fsync \
-inputDecodeBookFile=/home/rgruhn/ATRCOMPRESS/codebooks/codebook.cms56dec \
-outputFd=floatdata.fsync
```

  This will generate a file with 13 dimensional float framesync vectors. Both data and other frames (`TOF`, `EOF` etc.) are in the file.

- Q: I want to calculate CMS settings because I use a different microphone. How can I do that on the fly ?

  A: You can do that using your local client, by sending some RPCs.

```
$ATRSPREC/bin/ATRsend_RPC -host=localhost -rpcNumber=2 -receiver=ATRcontrol \
-command=WAKEUP
$ATRSPREC/bin/ATRsend_RPC -host=localhost -rpcNumber=2 -receiver=ATRwave2cep\
-command="CalcMeanPeriod=SPEECH"
$ATRSPREC/bin/ATRsend_RPC -host=localhost -rpcNumber=2 -receiver=ATRwave2cep\
-command="CalcMean=logpow+cep"
$ATRSPREC/bin/ATRsend_RPC -host=localhost -rpcNumber=2 -receiver=ATRwave2cep\
-command="ResetParamSumBuffer"
```

  Now you speak some utterances. When you finished, send the following RPCs:

```
$ATRSPREC/bin/ATRsend_RPC -host=localhost -rpcNumber=2 -receiver=ATRwave2cep\
-command="SaveParamMean=myfile.cms"
$ATRSPREC/bin/ATRsend_RPC -host=localhost -rpcNumber=2 -receiver=ATRwave2cep\
-command="LoadParamMean=myfile.cms"
$ATRSPREC/bin/ATRsend_RPC -host=localhost -rpcNumber=2 -receiver=ATRwave2cep\
-command="CalcMean=off"
```

  If you want this CMS file to be used from now on, change your configfile for SBpara, which is usually in:

```
c:/progra~1/python/sprecwin/win/config.sbp
```

  Please remember that updating the ATR-MATRIX client software may undo your changes or damage the config file.

# E Used Port Numbers

This chapter lists ports used by ATR-MATRIX. The numbers are mostly default settings, and can be changed at startup time.

## E.1 Server

On the server, the following ports are used by a ATR-MATRIX server:

**5560** ATRexec (the server side ATRSPREC) reads RPC commands from this port (= RPC number **5**)

**5598** `doubleserver.py` reads RPC commands from this port (= RPC number **43**)

**24042** on this port, ATRSPREC expects input data. If the server is configured to be used with a computer as client, then compressed cepstral data is expected. If the server is set to accept telephone clients, then raw ulaw data is expected. The data received on 24042 can be read from 24442.

**24142** this is the port for communication between CHATR and `doubleserver.py`. 16k 16bit (server's native byte order) wave signals or CHATR units will be written to this port.

**24242** this is where `doubleserver.py` awaits clients to connect. All clients connected to 24242 will receive any speech or unit data that CHATR sends to 24142, with a data header and eventually converted to a different audio type.

**24442** for debug purposes, the input data to ATRSPREC (either framesync ulaw speech or compressed cepstral data) can be read from this port. `atrOnlineDebug.py` can be used to display the cepstral data.

**(50000)** this port is where displaypow data is usually available. This is not used in server mode.

**(50001)** this port is where displaypow data is usually available. This is not used in server mode.

**50002** this port is where the client's GUI connects to. More than one GUI can connect to this port. The available data on this port are display commands. GUI commands can be sent to this port.

**50003** if started, `CPUuse.py` writes the current CPU load to this port. Per default this information is written to all connected sockets every half second.

## E.2 Client

Most port numbers on the clients are chosen randomly at startup time. The current settings are written to `stderr` by `loginclient.py`, but may become unknown under Windows as soon as the information scrolls out of the window.

**random** the RPC port of `loginclient.py` is set by random. It can also be set as command line option.

**random** the RPC port of `SBpara` is set by random.

**random** the RPC port of `chatrplayer.py` is set by random.

**(50000)** if set in the options of `SBpara`, displaypow data is available here. This data can be read by `MatrixGui.py`, if set in its configuration. Using displaypow is default for a Linux client, not using it is default under Windows.

**(50001)** if set in the options of `SBpara`, displaypow data is available here. This data can be read by `MatrixGui.py`, if set in its configuration. Using displaypow is default for a Linux client, not using it is default under Windows.

## E.3 Resource Manager

**5599** `matrixlogon.py` reads RPC commands from this port. `matrixlogon.py` is typically running on pcx196.

**42000** `matrixlogon.py` awaits login requests from clients on this port.

# F Data Formats

## F.1 GSM

Globale Systeme Mobile (GSM) is an RPE-LTP coder and operates at 13 kbps. It is widely used in Europe for mobile telephony. We downloaded and installed the following software:

| | |
|---|---|
| via | http://www.itl.atr.co.jp/comp.speech/Section3/Software/gsm06.10.html |
| details | http://www.cs.tu-berlin.de/~jutta/toast.html |
| source code | ftp://ftp.cs.tu-berlin.de/pub/local/kbs/tubmik/gsm/gsm-1.0.10.tar.gz |
| dos exe | ftp://ftp.netcom.com/pub/ne/neisius/gsmany09.zip |

## F.2 Example for CHATR Units

CHATR writes out a unit file, containing information about the wave files to concatenate and also several redundant information. For example, this is a CHATR unit file for the utterance "hai wakarimashita".

```
; Unit Stream plus
; (filename start duration num_units start_offset end_offset
;    (Seg_name source_dur target_dur)
;    ...)
; ...
(Utterance Unit
(
("/data3/atra51/itlusers/rgruhn/resources/chatr_dbs/MYS/wav/MYS_TRA_O_39.wav" 1430 233 3 0 0
   (  h   89   65) ;;        #ha   ; #haiw
   (  a  107   92) ;;        hai   ; #haiwa
   (  i   37   67) ;;        aiw   ; #haiwat
   )
("/data3/atra51/itlusers/rgruhn/resources/chatr_dbs/MYS/wav/MYS_TRA_L_24.wav" 1623 1301 13 0 0
   (  w  101   46) ;;        iwa   ; haiwaka
   (  a   50   91) ;;        wak   ; aiwakar
   (  k   75   83) ;;        aka   ; iwakari
   (  a   51   60) ;;        kar   ; wakarim
   (  r   29   33) ;;        ari   ; akarima
   (  i   49   49) ;;        rim   ; karimash
   (  m   55   51) ;;        ima   ; arimashI
   (  a   74   88) ;;       mash   ; rimashIt
   ( sh   67   63) ;;       ashI   ; imashIta
   (  I   46   32) ;;       shIt   ; mashIta#
   (  t   67   89) ;;        Ita   ; ashIta#
   (  a  119  111) ;;        ta#   ; shIta#
   (  #  518  600) ;;         a#   ; Ita#
   )
))
```

As we want to transfer only necessary data to keep the minimum bandwidth as low as possible, the unit file is refined with the program `chattr.py`. The example below represents the same utterance as the long unit file above.

Speaker followed by number of units in the utterance. Then, for each unit, a file ID, start point in seconds and duration in seconds is given.

```
mys 2
MYS_TRA_O_39 1.43 0.233
MYS_TRA_L_24 1.623 1.301
```

# G Client-Server Evaluation

This section contains additional information to the client-server ATR-MATRIX evaluation experiment.

## G.1 List of Tested Laptops

Table 9 lists the laptops tested for their sound device properties. As only few met the quality expectations for evaluation purposes, the sound device can be considered the weakest point in a laptop computer. Experiments also showed that unplugging the computer from electricity net and using battery power instead reduced the noise.

Table 9: A list of ITL laptops which were tested for their sound device quality

| IP address | Computer type | rating |
|---|---|---|
| itlnpc148 | Panasonic Lets note | no audio output |
| itlnpc131 | Panasonic Lets note | OK |
| twin52 | Panasonic Lets note | too noisy |
| itlnpc70 | Gateway 5150 | sound quality OK, but audio output is fed back into input |
| itlnpc72 | Gateway 5150 | too noisy and audio output is fed back into input |
| twin144 | IBM thinkpad | too small gain, too small output |
| itlnpc136 | Toshiba Dynabook | too noisy |

Especially interesting is that two Gateway Solo 5150 bought at the same time had different sound device quality. The one with the smaller serial number (i.e. the older one) had better sound characteristics.

## G.2 How To Do Evaluation

This section gives some hints on how to make evaluation. It was built on many mistakes that were made in this project, we hope that the reader will be able to avoid them.

Evaluation means to test your software on "normal people", i.e. volunteers who do not know your software and have to use it for you. Evaluation means aquiring data such as the recognition rate of a speech recognizer if used by naive users as well as information about user friendliness.

You will have to do the following steps:

**decide your goal** : This means e.g. to make a table with the data you want to have. Discuss this with the other people involved in your project, e.g. your supervisor. You may also want to read someone else's papers. You will probably want to get recognition rates, but you may also want to calculate other statistics like task completition rate or time taken.

**make a timetable** : Ask the various people you need when they have time for you. Avoid times with important demonstrations where everyone is busy. You need to have "naive speakers" as your test people, i.e. people who don't know your system at all. Sugaya-san can get such people for you, but it takes about one week to organize. It should be mentioned that for most experiments Melissa is hired as she is a very experienced English speaker whose voice is recognized very well. Hiring people costs money, which must be approved.

**setup your hardware** : Think of which data you will need to get the results you want. Discuss this with Ban-san. Usually the session is videotaped. Also, all audio is recorded on 4-track DAT tapes, with one track each for speech input and speech output on both sides, as this makes it possible to draw an exact timeline. You may wish to collect extra data, make a checklist and discuss it with Ban-san.

**test the setup** : Test your setup. Test all details, make a test session. Do not change anything after this test session, do not update your software. If you have to, repeat the test. In the real test you will have to give the test persons some written instructions, telling them which buttons to click and what to say when. Section G.3 is an example for a user manual. You also need some example schedules and hotel data, Sugaya-san can provide you with this. In most experiments, tests are performed with various Japanese speakers and few or only one English speaker. This allows you to test the scenario for words that are not recognized well for your English speakers. In general, the numbers "four", "five" and "seven" (as well as related words like "the seventh") are hard to recognize in English for some people. You do not want to test your Japanese speakers ahead for well recognized numbers, but it is recommended not to use 四日, 八日, 二十日, 六月, 四月 or 七月 in the hotel data and schedules. Also, 五 at the end of a number block is often misrecognized as 号.

**test session** : before starting, explain to the test persons one by one what you expect them to do. If those people are in Japanese, your instructions must be in Japanese and preferably given by a Japanese. Some people speak louder than others, adjust the volume settings for each person. During the session, depending on your setup, typically two people will talk without seeing each other. Ban-san can take care of the audio recording and mixer settings. There should be no one in the same room as the test person, if you want to watch the test person, you need to set up the video system accordingly. To ensure equal conditions for all test persons, you may want to shut down and restart ATR-MATRIX before each session.

**get recognition rates** : Several steps must be taken to get this data.

1. transcription. The one big speech file you have at this point is cut into small parts containing only the speech. Most of the audio data is silence, this is thrown away. This is usually done by the ITL1 labellers who also copy the audio data from DAT tape to file. Contact Naito-san for further details. For transcribing, about half a day per 1 hour of conversation (including silence) is needed. But as labellers may be busy, plan for 1 week waiting time until your job is scheduled to be done.
2. tagging. This is a morphological analysis of the speech, taking Kana or Kanji input and tag it as a noun, particle and so on. Contact Sumita-san for this work, it will take about 1 day per hour of conversation (including silence), and additional about one week before the work is started.

   Alternatively, morphologic analyzer programs can do it much faster but not 100% reliable (estimated 95% accuracy). The following scripts have to be started for automatic analysis of speech and generation of so-called *answer files*, assuming that `$DATABASEROOT/LNG/JTEXT` is the place where the transcription files (`.JTEXT`) are saved:

   To check JTEXT format

   ```
   $ATRSPREC/script/checksdb/checksdb -jtext -jtextdir $DATABASEROOT/LNG/JTEXT
   ```

   Generate JMOR from JTEXT (works only on DEC-Alpha or Sun).

   ```
   $ATRSPREC/script/dictools/jtext2jmor.pl $DATABASEROOT/LNG/JTEXT/*.JTEXT $JMOR_DIR
   ```

   or, in case you need only one side of the dialog,

   ```
   $ATRSPREC/script/dictools/jtext2jmor.pl -oneside /DB/MDB/CDB08/LNG/JTEXT/*.JTEXT $JMOR_DIR
   ```

   Generate ANS from JMOR

   ```
   $ATRSPREC/script/dictools/jmor2ans.pl -unit utterance $JMOR_DIR $ANS_DIR
   ```

   Tsukada-san can help you if problems occur with the scripts.
3. recognition. If the transcription and tagging are provided, the recognition rate can be calculated. The program to be used for this is `$ATRSPREC/script/atrLatt.py` As parameters, this script requires the config file used for ATRSPREC in the evaluation, an ascii-file with the speaker ID and file names, the answer file type `ansSpecial` of your answer files (ask Singer-san) and information about your database. It is recommended to look at the option section of the script code. A possible command line is:

   ```
   atrLatt.py -config=config.matrix -AsciiFile=/DB/MDB/CDB08/INFO/ascii.info  \
   -calcCMS=CalcMeanPU -ansSpecial=EDB                                         \
   -db="{'wavDir':'/DB/MDB/CDB08/SPH/WAV/JAPANESE',                            \
         'wavExt':'16k',      \
                 'trsDir':'/DB/MDB/CDB08/SPH/TRS/JAPANESE',                    \
                 'trsExt':'TRS',      \
                 'ansDir':'/RR/Recognition/CstarIIJ/ldata/19990831/CDB08/ANS',\
                 'ansExt':'ANS',      \
                 'latDir':'Lattices01',      \
                 'latExt':'LAT',}"          \
   -mrgFile=$MATRIX_RESOURCES/sprec_j_model/lmodel/19990614/merge12.list
   ```

   Writing the command and the command line options into the config file makes a redo of the experiment easier

## G.3 User Manual

# ATR-MATRIX による会話実験

会話に関する説明

顧客側

本実験にご協力いただき、ありがとうございました。本実験の設定は次の通りです。

- アメリカに旅行する予定であるが、英語を話すことができない。
- ニューヨーク・シティ・ホテルの部屋を予約しようと思っている。
- ノートPC（ラップトップ）を持っており、インターネットを使って予約をしようとしている。

まず、次の注意と別紙にあるスケジュール等の情報をよく読んでください。

私たちは、できる限り現実的な会話を収集したいと思っています。全てのハードウエアは、こちら（ATR のスタッフ）で準備しますが、システムの起動やインターネットへのモデム接続は各自でお願いします。

本番の会話に入る前に、次を実行してください。

1. マウスに慣れる： ラップトップの下側のボタンはマウスの左ボタンに、上側のボタンは右ボタンに対応しています。
2. インターネットに接続する： 「Modem connection」アイコンをダブルクリックしてください。ウインドウが開きますが、ただ「接続」ボタンをクリックしてください。（番号、名前、パスワードは変更しないでください。） 接続が成功すると、そのウインドウは消滅します。
3. ATR-MATRIX を実行する： 「Startmatrix」アイコンをダブルクリックし、しばらくお待ちください。ATR-MATRIX ウインドウが開きます。
4. ATR-MATRIX をホテルの予約係に接続する： ATR-MATRIX ウインドウの上部のメニューバーには「file」（左上）があります。その「file」を左ボタンで一回クリックしてください。メニューが表示されますので、その中の「connect」をクリックし、新しいウインドウが開かれます。下の方に表示される「OK」をただクリックしてください。

以上で準備は終りました。ATR-MATRIX を使ってホテルの予約係と話すことができます。

「WAKEUP」ボタン（左上）をクリックすると、システムが動き出します。中断したい時には、「GOTO-SLEEP」ボタン（左上）をクリックしてください。

ATR-MATRIX ウインドウの右上には、システムの稼働状況を表すレベルメーターが表示されます。レベルメーター中の赤い部分が長い時には、計算機が混んでいますので、少々お待ちください。その赤い部分が短くなってから、次の発声をしてください。

# ATR-MATRIX Conversation Experiment

Explanations for conversation
Customer part

Thank you very much for taking part in this experiment.
We ask you to imagine the following situation:

- You plan to travel to the USA and speak no English.
- You want to reserve a room in the New York City Hotel.
- You have a laptop and want to do this reservation using the internet.

Please first read this and the following (Japanese) pages with information about your time schedule and some other data.

We want this scenario to be as close to reality as possible. All hardware will be set up by ATR staff, but you will have to do all perparations a real user would have to do, too. This includes connecting the laptop to the internet with a modem.

Before beginning the actual conversation, please

**get used to the mouse handling** : On the Laptop used, the bottom button is used as "left mouse button", the upper one is the "right mouse button".

**connect to the internet** : doubleclick on the icon `Modem connection`. A window opens asking for number, username and password. They are already set, so just click on 接続 (connect). When the connection window vanishes, the connection is established.

**start ATR-MATRIX** : doubleclick on the icon `Startmatrix`. Please wait a moment until the ATR-MATRIX GUI window opens.

**connect your ATR-MATRIX to the hotel reservation service** : On the top of the ATR-MATRIX GUI window you will find the menu item `file`. Select it by clicking on it once with the left mouse button. Then, in the appearing menue, select `connect`. A window opens and asks for connection settings, please just click `OK`.

Now you are ready to talk to the hotel staff using ATR-MATRIX.

Please press the `WAKEUP` button to activate the system and the `GOTOSLEEP` button to pause, e.g if you want to talk to the administrators.

In the upper right corner of the ATR-MATRIX GUI window there is a meter for the CPU usage. When the red bar is long, the computer is busy. Please wait until it becomes short again before saying the next sentence.