Signal processing via web services: the use case WebMAUS

Thomas Kisler[1], Florian Schiel[1], Han Sloetjes[2]

Institute of Phonetics and Speech Processing[1], LMU München,
Max-Planck-Institute for Psycholinguistic[2], Nijmegen
{kisler,schiel}@phonetik.uni-muenchen.de[1], han.sloetjes@mpi.nl[2]

Abstract

The CLARIN infrastructure aims at providing a technical infrastructure for language resources. In this context we present the design of a web service and a corresponding interface to provide easy access to an formerly only locally executable application for automatic segmentation and labeling called Munich AUtomatic Segmentation (MAUS). Using a standard description format we not only make the manual usage of those services possible, but also enable access from within other services or chaining engines like WebLicht. As an example for the integration of the web service in another application, we show the WebMAUS integration into ELAN.

1. Introduction

The working draft of the HTML5 standard had a big influence on the possibilities of web interfaces and applications. Though almost everything has been possible before, the new standard makes many things much easier and defines functionality in a way that all browsers could implement it. And especially the new standard and the big hype around web 2.0 in general changed the visibility of web applications.
Services commonly referred to as web services are a convenient way to make software applications freely available without the hassle to install software on the local computer. Applications under development may be tested by a broader user community at an early stage (alpha) without the usual logistic problems connected with early version testing. Furthermore legal issues regarding the copyrights of our application code can be solved easily when the application stays hidden from the end user. Web services may be used by individual users via appropriate web interfaces or in batch mode from the command line or by other applications calling the web service as a "helper application".
In this contribution we present a first example of a linguistic web service that automatically segments and labels a spoken utterance into its phonemic contents, implemented as a CLARIN[1] conform web service, as well as examples for a web interface and the usage as a helper application to the well-know ELAN tool (**?**).

2. Munich AUtomatic Segmentation (MAUS)

Language resources often contain some kind of speech recording, either as a single or multiple channel sound file or as the soundtrack of a video recording. Many analyses require some kind of alignment of a symbolic representation (annotation) of the recorded speech act (i.e. an orthographic or phonetic transcript) to the corresponding parts of the speech signal. Such an alignment is usually called a segmentation and labelling (S&L). While the orthographic transcript of the truly spoken content can be obtained with the aid of transcribers (sometimes even via mechanical turk) at moderate effort, the S&L at word level or even worse the S&L on smaller linguistic units such as morphs, syllables or phones is expensive, time-consuming and error prone.
There are several ways to automate the S&L process. A simple way is the so called forced alignment to a given sequence of phones using speech recognition technology such as Hidden Markov Modelling (HMM). Here the aligner has the task to find the best partitioning of the speech signal given a fixed sequence of phonetic symbols and a set of pre-trained statistical models for each phoneme class of the language. Forced alignment works very well granted that the signal is of moderate good quality and the truly spoken phones are known a priori (which is usually not the case because given the spoken words usually only a citation form of pronunciation can be calculated automatically which deviates from fluent spoken speech).
The Munich AUtomatic Segmentation system (MAUS) extends the basic aligner concept by modelling a statistical space of possible pronunciation variants for a given orthographic input (**?**; **?**). Figure 1 shows a simple example for the german word 'Abend'. The hypotheses space is calculated for each individual text input based on a machine-learned statistical expert system of pronunciation. Combined with HMM technology the MAUS can thus not only find the best segmentation but at the same time the most likely sequence of truly spoken phones in the speech signal (see Figure 2 for MAUS S&L result as viewed in praat).
On a subset of spontaneous German speech in the Verbmobil corpus (**?**) the MAUS technique yielded about 97% of the average interlabeller agreement of three trained phoneticians working on the same task (**?**).
MAUS is implemented as a system of UNIX script files and C++ binaries that can be run on Linux and Windows platforms. It requires as input the speech signal and some form of either orthographic or phonological transcript of the spoken utterance. The result is stored in either BAS Partitur Format BPF (**?**), praat TextGrid[2] or Emu (**?**) compatible annotation format files. MAUS currently (version 2.30) supports 7 languages: German, English, Hungarian, Italian, Estonian, Spanish and Dutch (the first 4 including tokenizing, text normalization and text-to-phoneme conversion). A vast number of options allow the user to control

[1] http://www.clarin.eu/

[2] http://www.praat.org/

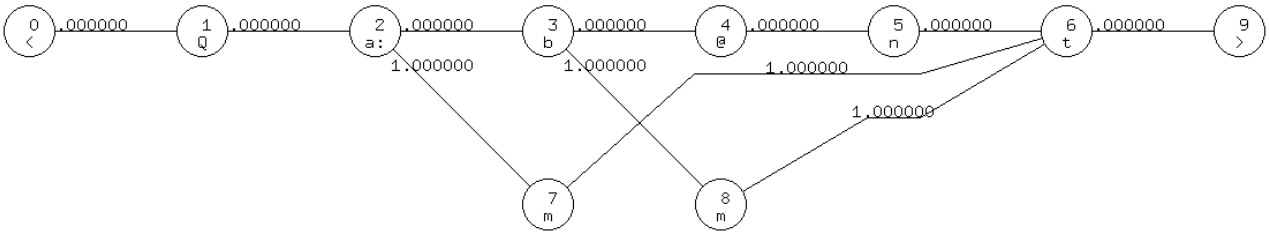

Figure 1: Example of a pronunciation hypotheses space for the German word 'Abend'.

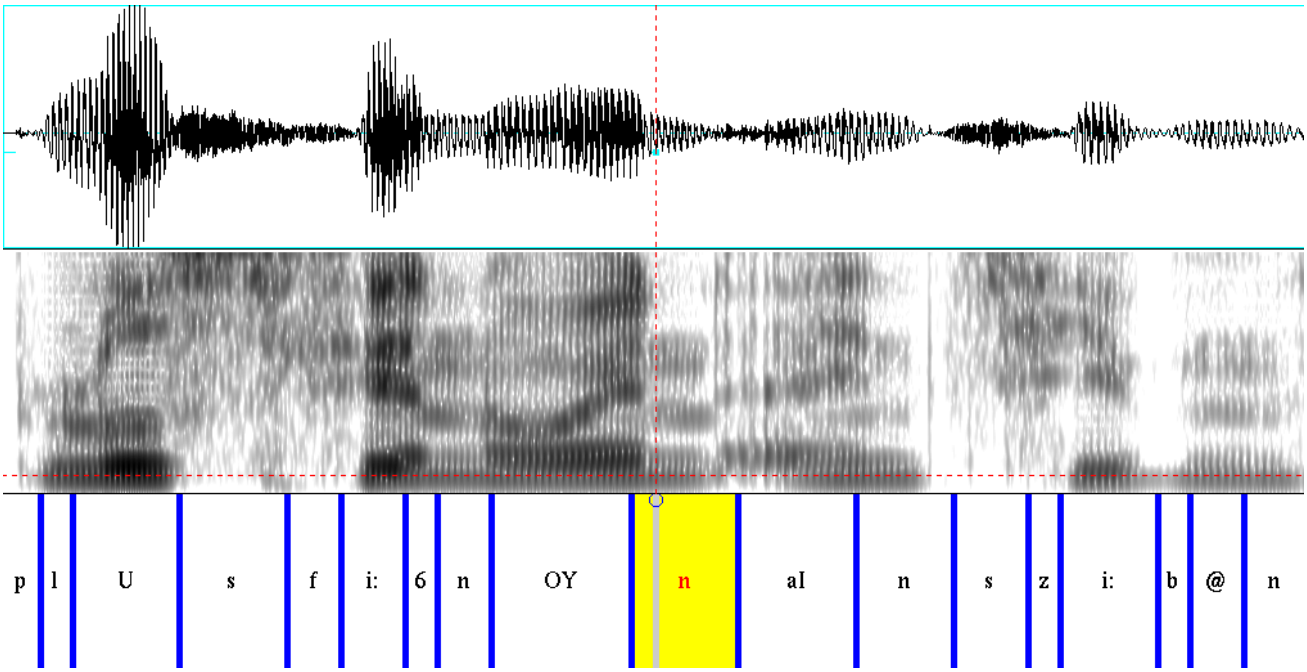

Figure 2: Example of a segmentation & labelling created by MAUS.

the S&L process as well as the form of output formatting and the statistical modelling of the pronunciation variation (e.g. by learning new pronunciation models or formulating explicit phonological pronunciation rule sets). The MAUS freeware package can be downloaded from the Bavarian Archive for Speech Signals[3].

3. WebMAUS Services

As a contribution to the "Common Language Resources and Technology Infrastructure" (CLARIN) project, we developed a web service that provides the functionality of the aforementioned MAUS tool. This web service allows access to MAUS without the need of installing it locally on the users machine, might the "user" be a human or another web service that wants to access it. The CLARIN view on web services is rather a process oriented view, especially when regarding the efforts that are put into implementing work flow engines, that allow the chaining of different web services. That was one of the main reasons we decided to implement the web services as RESTful remote procedure calls (RPC).

We wanted to use standard technologies to provide an easy-to-use and easy-to-understand interface to our web services. Therefore some of the HTML operations serve as an envelope for the data necessary to our web services. Through the "overloading" of POST we can achieve customized behaviour without breaking the RESTful idea (**?**).

To provide a machine readable format, we described our web services in the standard description format for RESTful web services, the Web Application Description Language (WADL). The WADL contains the technical aspects of a certain web service and therefore allows programs or chaining engines to generate the call to the WADL described web service automatically.

In addition to the technical WADL description, the CLARIN meta data infrastructure (CMDI[4]) format gives us the possibility to describe our web services also semantically. Each web service is described in the CMDI file, based on components that are registered in a public component registry. The harvester for CMDI metadata that are built to harvest the metadata of data can then also be used to harvest the metadata about the available web services in the CLARIN infrastructure (**?**).

So far we provide a variety of different RPCs that offer access to two MAUS modes, a simple basic and a general functionality. The "runMAUSBasic" takes a plain txt and a

[3] http://www.bas.uni-muenchen.de/forschung/Bas/software/

[4] http://www.clarin.eu/cmdi

signal file as input and returns a Praat compatible TextGrid file for a number of languages[5]. In this case the web service is extended by a language specific tokenizer and text normalization followed by a statistical-driven text-to-phoneme algorithm (Balloon (**?**)) which converts the arbitrary input text into an ordered sequence of canonical pronunciations. Other than the language no parametrization is allowed. The more powerful "runMAUS" call requires a signal file and a BAS Partitur Format file (BPF) containing a tier with already tokenized and phonemic encoded speech. In contrast to the simpler "runMAUSBasic" this web service allows the caller to provide the full set of optional parameters of the MAUS tool. With those parameters the MAUS call is fully customizable and therefore appropriate for advanced users. By incorporating both calls into simple script loops vast amounts of data can be processed automatically (batch processing). Other web service calls provide documentation and information about language specific phoneme sets used by the MAUS services.

The beforementioned descriptions can then be used to manually or automatically integrate WebMAUS to any infrastructure that also understands these descriptions. Example for those Service Oriented Architectures (SOA) are WebLicht and Taverna.

An integration of WebMAUS into the WebLicht tool chain (**?**) TODO Weblicht is already planned. WebLicht is a chaining architecture to combine many different processing steps to answer various research questions. This tool chain profits from two mentioned advantages of such a Service Oriented Architecture, which a that no tools have to be installed on the users machine and that it can access functionality where the source code might not be available. The tool chain was build in an earlier project (TODO cite weblicht) and was designed to process Linguistic data. The biggest problem with an easy integration of multimodal data so far is the exponential bigger amount of data that has to be processed. The current idea in WebLicht of piping the complete data through the chain is probably not applicable anymore. Though making the data, that should be processed, available via a weblink raises the problem of persistency and data storage. So far the architecture was based on the assumption that one file contains both the source and the result of a processing step. This becomes obviously unapplicable for the input of multimodal data, as in addition to the symbolic data also the signal/video files have to be available. But also for the output this creates the challenge of either adapting the current file format, so that results of signal processing can be integrated or additional files containing those results have to be piped through the chain.

4. WebMAUS Interfaces

As a show case for our developed web services and as a user interface for technical unfamiliar users we developed a few state-of-the-art web interfaces.

A web interface should provide an intuitive, natural-web-compliant way of processing the data. We based the interface on recent technologies like HTML5 and jQuery. The working draft of the HTML5 standard provides a variety of features that emerged from the widespread use of HTML and were missing in older standards. Those are audio players, advanced forms and also desktop-like behaviour, like drag and drop (**?**). jQuery, as an abstraction to Javascript, hides much of the counter-intuitive aspects of Javascript for the developer and provides a rich set of functions for GUI design and Ajax (Asynchronous Javascript and Xml) calls.

A big advantage of those web technologies is, that they can be run from within every standard compliant browser. Unfortunately, HTML5 is no standard so far and only a handful browsers implement the working draft HTML5 in a broader sense. It is to hope that once it is released that future generations of browsers will implement the standard in a more compatible way, than this was the case the past decades. Until now we propose to use Mozilla Firefox or Google Chrome, since both of these browsers implement a big subset of the specifications of the upcoming HTML5 standard. Both of them are freely available and Firefox furthermore is open source, so that theoretically everybody has access to our web interfaces.

As a demonstration for our "runMAUSBasic" and "runMAUS" functionality, three interfaces have been developed[6]. For the "Basic MAUS" interface only a plain text file and a signal file with the utterance are necessary. This is the easiest to use interface and useful for single signal and text files. The visual feedback presented to the user is responsive and entertaining, and results can be stored in a widely used standard format (Praat TextGrid). User may select between different languages, for which the BPF generation is implemented, since text normalisation and grapheme to phoneme conversion are already available. The second interface "General MAUS" gives access to the general MAUS interface, takes a signal file and a BPF file as inputs, and provides a set of options to the user which he might change to achieve better results. The user may choose between 7 languages and three different output formats[7]. The last web service, and the most interesting one especially for processing large amounts of data, is "Multiple MAUS" (see Fig. 3). It not only allows the full customization as does the general interface, but also for uploading several files via drag and drop. The combined size of files is limited to 300 MB and the interface has been successfully tested with a set of 600+ files (300+ signal-text pairs).

5. Calling WebMAUS from ELAN

ELAN[8] (**?**) is an audio and video annotation tool for the desktop, that is being developed by The Language Archive department of the Max Planck Institute for Psycholinguistics. It is applied in several types of research within linguistics and beyond. Annotation in ELAN is still mainly a manual process and although it offers specialized user interfaces for the before mentioned steps of segmentation and labelling (S&L), producing fairly accurately aligned, multi-layered annotations to the primary data, is time consuming (and therefore expensive). In recent years first steps have been

[5] currently English, German, Hungarian and Italian

[6] https://webapp.phonetik.uni-muenchen.de/BASWebServices

[7] BPF,TextGrid and Emu.

[8] http://www.lat-mpi.eu/tools/elan/

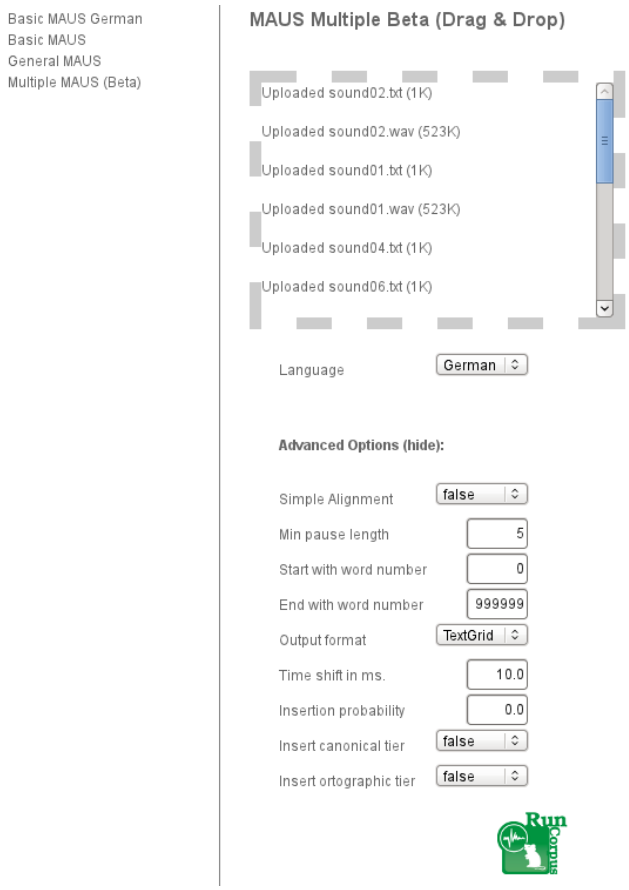

Figure 3: "WebMAUS Multiple" screen shot with advanced options.

taken to extend the tool with software components that semi-automatically segment and/or label media signals. A language independent silence recognizer was the first of such extensions. Next, in the AVATecH[9] (**?**) project the extension mechanism for recognizers was enhanced in order to accommodate extension with components created in a variety of programming languages (ELAN itself is written in the Java programming language). Because the recognizers in AVATecH run on different platforms, it seems natural to make them available as web services and that is what currently is being worked on.

The initial implementation of the interaction with WebMAUS builds on this existing extension mechanism of ELAN (therefore it is not an integral part of ELAN but a separate module). The advantage of this approach is that the WebMAUS client software can be replaced easily and repeatedly independently of the update cycles of ELAN itself. In this first implementation the "runMAUSBasic" variants for four languages are supported. The information about the web services and how to invoke them, is extracted manually from the WebMAUS WADL file and is stored in a meta data file describing the module and its parameters. Whether it is possible to build a generic web service client module that can invoke any web service that's sufficiently described by a CMDI and a WADL file, is still unclear and to be investigated. Interaction with WebMAUS, at this stage, consists of uploading a wav file and a text (or text file) to the web service and converting the returned content to tiers. The user interface of ELAN allows the user to either select a text file from the file system, or to select a tier for upload. In the latter case ELAN converts the contents of the tier to plain text and uploads that to the web service. The results are currently returned as Praat TextGrid files.

This scenario is an example of bringing the data to the algorithm and for each call the data is uploaded again. Since the "basic" WebMAUS does not support parameters to configure the workings of the service, it is not likely that the same file needs to be uploaded more than once (the result will always be the same). But for more complex processing it would be beneficial to upload the audio separately and then supply the URL of the uploaded file to the web service.

As possible enhancement of the WebMAUS client extension, supporting the general "runMAUS" service can be considered. For that purpose ELAN should ideally be able to produce BPF files. A further enhancement can be reached in post processing the results. While the fine segmentation is advantageous in many cases, it is often also required to have aligned annotations on the level of bigger units e.g. "sentence". Joining smaller segments to build larger units (as a separate tier) could be a next step to improve the usability of the MAUS technology for ELAN users.

6. Conclusion

The described web services and interfaces have reached a stable state and are now used for the research and teaching at the "Institute of Phonetics and Speech Processing" in Munich. Especially the much more eye-candy and, for users who are familiar with the web, intuitive interface seemed to have led to an improvement in the user experience. Even lower semesters who are not yet used to to handle the tools and programs that are commonly used within the research community have been able to produce $S\&L$ data after been shown the interface once.

The WebInterface furthermore saves the hazzle of local installation and permanently updating it and makes the tool available to a broader audience, which are not able to install a set of tools that are supposed to interact which each other. People that are familiar with a command line interface can profit from the possibility to call the web services from the command line itself or scripts over tools like curl[10].

Researchers and students that are already used to tools like ELAN can profit from the easy integration of web services into their favorite GUI. As the interface is clearly structured the integration of a web service is much easier than integrating functionality by other means (e.g. through libraries). The developers of tools, too, profit from the fact that no updates of the provided funtionality on the client are necessary. In a well-designed user interface additional work might be taken from the user, as it can be seen in ELAN. There the user not only is able to select a file from the filesystem, but also send a tier to the web service which might save him some time in creating such a text file.

[9] http://www.mpi.nl/avatech

[10] http://curl.haxx.se

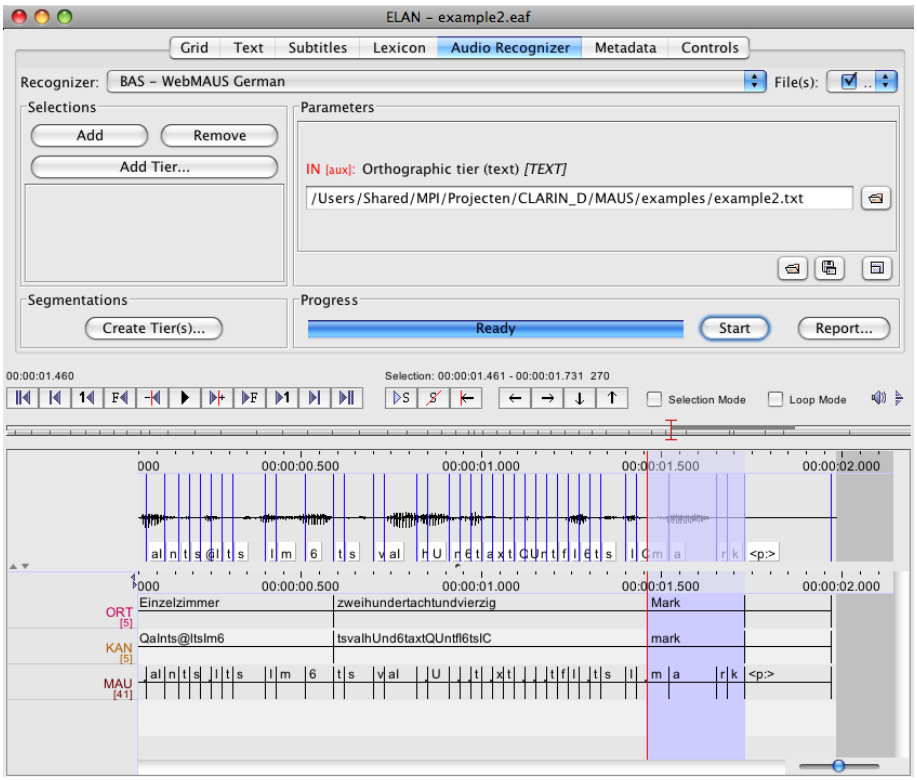

Figure 4: The results of a call to WebMAUS, three tiers have been created.

7. Acknowledgements

The work of the authors was carried out within the CLARIN-D projcect (**?**) (BMBF-funded).