Framework For Consistent Speech Databases

Sören Wittenberg, Rüdiger Hoffmann**

*Institute of Acoustics and Speech Communication, Technische Universität Dresden , Germany

Soeren.Wittenberg@tu-dresden.de, Ruediger.Hoffmann@tu-dresden.de

Abstract

The introduced speech processing framework creates phonetically and prosodically annotated speech databases. It provides with structured data files in eXtensible Markup Language format. Those files include all available information about a recorded utterance inclusively the speech signal. A Document Type Definition (DTD) describes the data structure and provides with the possibilty of automatically data structure validation. That ensures right data reading by human and interoperability between the used speech processing tools. A user can browse the speech databases with a normal web browser. The browser has to support XSL transformation, ECMA script and Scalable Vector Graphics to visualize the content. If the user requests a utterance, the browser gets the requested file with all available information of the corresponding utterance. The advantage of that is that the user obtain same data as a speech processing tool when it uses the underlying file server. The navigation through different speech layers is like browsing a web page. The user clicks on a part he wants to look at and a file embedded ECMA script filters the data and modifies the screen. The script is part of the XSL transformation file. It allows also elementary editing of the utterance content like changing word boundaries by moving the corresponding boundary mark. The changes are committed to the web server, that can handle further processing like integration into a subversion system. Because the whole speech database contents are strings, a standard search engine can be used for database searching. Searching for a phoneme under special context yields to all avialable phonemes with all information of them and also the speech signal.

Index Terms: speech database, framework, structured data

1. Introduction

Speech database improvement is very important to increase the quality, usability and performance of a database. Development of new speech annotation algorithms make it necessary to replace parts of an existing database by new, and hopefully more precise, annotations. A speech database contains different information layers like a signal, phonetic, morphemic, prosodic, syntactic and an orthographic layer. Depending on the database implementation goal, the layer contents are stored in different files as shown in the block diagram 1 of our old speech framework [1][2]. It is subdivided into four blocks. While data collection is providing the raw material for the annotation, the phonetic and linguistic processing blocks generate the required signal segmentation, transcription and prosodic annotation. The information combination step collects all data and provides with different target data files. The resulting speech database quality is very good, but the access to the different speech information layers is poor. Handling with different files can provide with advantages in the further processing cycle, but when the data format differs from file to file it becomes difficult to interpret the data right. Furthermore, the user has to know, which data can be processed which way. Speech, as well as music, is highly structured and that implies the usage of a structured data definition to describe the speech layers. A widely used technique to describe structured data is using *eXtensible Markup Languages*. The advantages of XML are a precise definition of the data structure, human readability, ability of automated structure validation and processing. Because of these advantages, it was the aim to modify the speech processing framework, the structure of the created speech database and also the access to the speech database. To obtain the given aims, the framework provides with only one structured data file per recorded utterance.

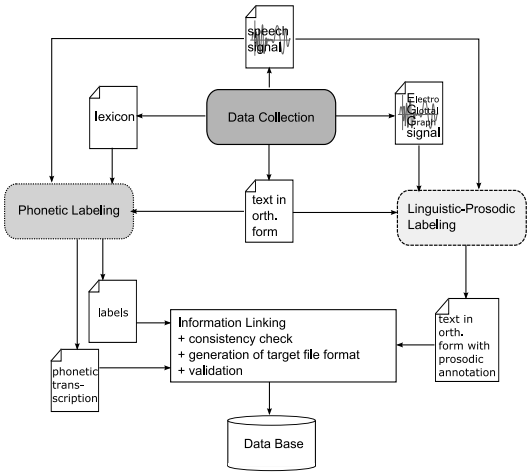

Figure 1: Typically structure of a framework for high quality speech databases creating (according [2]).

2. Speech Processing Framework

Our speech processing framework was first established at [1]. It rests upon the system which creates the TC-STAR database [2] and includes all necessary algorithms, procedures und rule-formulated expert knowledge for creating an annotated speech database. The framework is a modular system corresponding to speech layers. The replacement of a module within a layer is also possible like replacing the whole layer. Because it was not the aim to use only proprietary development, the integration of foreign components became a challenge we solved by using a wrapper application, that transforms the frameworks XML based data exchange format into an application in- and output data format. Furthermore, we decide to change the

parallel framework design presented in figure 1 into to a serial design (figure 2). The loss of parallel layer computing is being replaced by the weightier advantage of a more reliable interoperability between the used modules. The necessary information combination module was eliminated and only the data validation module get place at the new system. Foreign modules, which do not support the used data exchange format, will be encapsulated into the wrapper application. That ensure the interoperability to the foreign modules. Interoperability between own modules is ensured by an obvious data description based on Document Type Description (DTD). That will get speech databases with a consistent structure and formal regularity, that can be automatically validated.

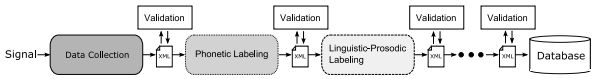

Figure 2: Serial structure of the more flexible speech processing framework.

In the area of speech signal processing, it is generally accepted that markup languages provide with a lot of advantages. There are many frameworks, annotation and processing systems are able to processes structured data. There are also a lot of recommendations from the W3C describing different speech processing tasks. The most noted speech processing markup languages are *Voice Extensible Markup Language* (VoiceXML), *Speech Synthesis Markup Language* (SSML) and *Pronunciation Lexicon Specification* (PLS). *Speech Recognition Grammar Specification* (SRGS) and *Semantic Interpretation for Speech Recognition* (SISR) are two other examples, which are less used. But the recommendations are not granular enough for most applications. One example is the well-known TTS system Mary [7]. Inside the system there is a XML based data representation format, which is called MaryXML. This ML is extented by many entries required by the TTS system. These entries are not part of the Speech Synthesis ML.

Our speech processing framework extends and combines the known speech processing markup languages also. This ensures a high granularity speech data annotation whilst the abidance of W3C recommendations enables an integration toward foreign tools. Speech data annotation is divided into two description layers. First layer uses SSML to make a coarse orthographic and prosodic description of the whole utterance. Pauses, accents and phrase boundaries can be marked by tags like <*break time="220ms"*> to denote a 220 ms long speech pause. This layer is redundant and can be generated from the second layer, but the abidance of the SSML recommendation ensures the speech synthesis by a SSML conformal TTS system.

The second layer provides with a more accurate speech data description respectively transcription. The following listing shows a part of the speech database entry for the utterance *Parliament budget ought not ...* . The first layer is denoted by the tag <*speak*> while the second part is the rest of the listing.

Listing 1: Part of a speech database file.

```
<?xml version="1.0" encoding="UTF-8"?>
<prophano version="1.0" xml:lang="en-GB" ...>
  <speak version="1.1" xml:lang="en-GB" ...>
    <lexicon uri="file:///lexicon_laura.pls" .../>
    <voice name="Laura" gender="female" age="32">
      <lookup ref="lexicon_laura">
        <p>
          <s>
            <emphasis level="moderate">
              Parliament
            </emphasis>
            budget
            <break time="220ms"/>
            ought
            ...
          </s>
        </p>
      </lookup>
    </voice>
  </speak>
  <voice name="Laura" gender="female" age="32">
    <p>
      <s>
        <pros emphasis_level="moderate">
          <w start="0.06s" end="0.60s">
            <orth>Parliament</orth>
            <phon>
              <syllable start="0.06s" end="0.42s">
                <phoneme start="0.06s" end="0.15s">
                  p
                  <audio encoding="base64Binary"> ...
                </phoneme>
                <phoneme start="0.15s" end="0.42s">
                  A:
                  <audio encoding="base64Binary"> ...
                </phoneme>
              </syllabe>
              ...
            </phon>
          </w>
        </pros>
        <pros break_time="220ms">
          <audio encoding="UTF-8"> ... </audio>
        </pros>
        <w start="0.60s" end="1.14s">
          <orth>budget</orth>
        ...
</prophano>
```

First layer does not describe the utterance phonetically. Therefore, SSML defines a entry <*lexicon*> to reference to external pronunciation lexicons. A lexicon entry <lexeme> contains the pronunciation of a word including pronunciation variants inside the <phoneme> tags. The structure of the lexicon is defined by the W3C recommendation *Pronunciation Lexicon Specification*. The PL-Specification claims the use of a *IPA* alphabet, but we use *eXtended Speech Assessment Methods Phonetic Alphabet* [3] to obtain an easier ASCII presentation and processing. Following listing shows one lexicon entry. The second layer does not need this lexicon, because it includes the phonetically description. That is also the reason a special utterance lexicon can be created by the second layer and used by the first one.

Listing 2: A lexicon entry

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0" alphabet="x-sampa" ...>
  <lexeme>
    <grapheme>leg</grapheme>
    <phoneme>leg</phoneme>
  </lexeme>
</lexicon>
```

2.1. Audio Signal

The aim of the speech processing framework is to provide with only one structured data file per recorded speech signal.

	Music	Noise
sample frequency [kS/s]	44.1	48
Bit per sample	16	16
length [min.sec]	5.25	5.00
samples	14,314,752	14,400,000
MSWave file size [MB]	54.6	27.4
UTF-8 size [MB]	167.4	94.6
UTF-8 size gzip [MB]	73.8	37.1

Table 1: Comparison of the file size when storing a music signal (-12 dBFS RMS, two channels) and white noise (-8 dBFS RMS, one channel) as gzip compressed UTF-8 files.

Therefore, audio samples are integrated into the XML file. The element <audio channel=..> marks the audio signal of different audio channels. This entry is always placed inside the innermost description level. If the innermost level is the phoneme, than it is placed inside the entry <phoneme>. If the innermost level is the syllable, than it is placed inside the entry <syllable> and so one. The reconstruction of the whole audio signal is made by collecting all entries <audio> according their position inside the document and concatenate all together. Audio samples can be stored as *base64Binary* data, but it should be prefered to store each sample as plain UTF-8 strings. The resulting storage amount will be reduced using a gzip compression over the whole document. Table 1 give an example when storing two different signals as UTF-8 and as gzip compressed UTF-8 strings.

2.2. Database Access

The speech database is organized in folders and files. Each file contains one speech recording inclusive all descriptions and information about the recording that are available. The XML files are compressed by using gzip. The uncompressed files are encoded with UTF-8. Access to the database is available via file or HTTP server as depict at figure 3. User restricted access via HTTP server is gainful because the server communicates the content encoding type (gzip) to the requesting client. If the client is a web browser, it will uncompress the file and processes the including content without futher processing steps by the user. That mechanism ensures a world wide high performing database access.

3. Monitoring, Visualization and Editing

3.1. Self-Monitoring

A validation module ensures the formal correctness, data integrity of a given XML file by using a global defined *Document Type Definition*. The DTD is a special file that determines the formal structure of a XML document and consists of declarations for elements (<!**ELEMENT** ... >) and element attributes (<!**ATTLIST** ...>). The element declaration defines the element name and the content type. The content type is given by round brackets and could be other elements (childs), a sequence of elements or strings. Element attributes can be defined by the ATTLIST tag. The occurence of an attribute is denoted by #REQUIRED or #IMPLIED and the occurence of childs by + (one or more times), * (any number of times including zero) and ? (zero or one times). Listing 3 shows a part of a DTD, that specifies the element <phoneme> that

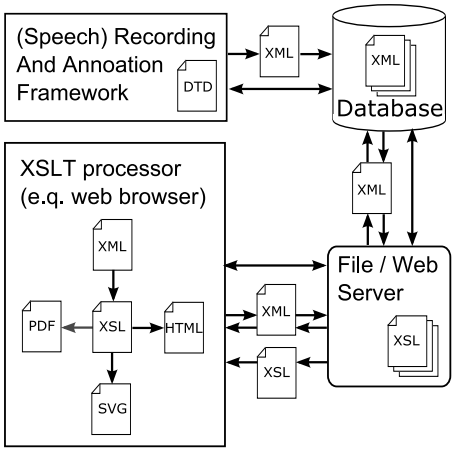

Figure 3: Main concept of speech database access.

can have an child element <audio> and must has a string (phoneme symbol). The required attributes are *start* and *end*. Both are of type character data. The third attribute is the hash value that is only optional.

Listing 3: Part of a DTD

```
<!ELEMENT phoneme (#CDATA, audio?)>
<!ELEMENT audio   (#CDATA)>
<!ATTLIST phoneme start    CDATA #REQUIRED
                  end      CDATA #REQUIRED
                  hash     CDATA #IMPLIED>
<!ATTLIST audio   encoding CDATA #REQUIRED
                  channel  CDATA #REQUIRED
                  hash     CDATA #IMPLIED>
```

Creating a DTD is easy when using the class diagramm approach of the Unified Modeling Language (UML) as a visual structuring facility. The transformation of an UML representation into a DTD is made rule-based [4].

After one component has been processed a XML file, the validation module runs and validates the formal structure of the document. The module also checks and stores a hash value that is made by the Adler-32 checksum algorithm for major blocks. A major block, for instance a sentence with all child elements, can be validate calculating the CRC and comparing with the stored one. If there is no different, the module will not check the inner block structure, otherwhile it validates the block structure. The Adler-32 checksum algorithm is used because of its simplicity and the good performance.

3.2. Visualization

The concept of the framework respectively a finished database is *click and see*. The user has not known anything about the available data of a recorded utterance. He will click on a file and he gets all available data inside one structured data file. But against others (e.q. [5] or [6]), we decided to transform the content client side. If the user wants to store and process the requested utterance by his own tools, he can do it. The visualization of the file content can be made by a normal web browser. Todays browsers are powerful enough to transform the content of the XML file into a text and also a graphically representation like it is needed to visualize the audio signal

with all annotation marks. The *eXtensible Stylesheet Language Transformation (XSLT)* is part of the markup language family *eXtensible Stylesheet Language (XSL)*. It can be used to transform an XML document into a holy different target language structure like xHTML, SVG, VoiceXML, ASCII, PDF and others more. The transformation is made by a XSLT-processor. Such a processor is part of nearly all current desktop web browsers. In our case, the browser transforms the structured content into xHTML code with embedded SVG and displays the result on screen. The transformation rules are defined by a XSL transformation file. To increase the capabilty of the XSL transformations the so called XPath technology can be used to filter data from the XML document while transforming. These become imported when only parts of the utterance should be extracted (e.q. all phonems 'a' or all realizations of the word 'is').

The result of a transformation is shown in figure 5. The area I shows information about the audio signal. Area II is the text representation of the audio signal and area III the correspondig graphically representation as a Scalable Vector Graphic. The figure marked by IV is a zoom into to signal using the zoom functionality of the browser.

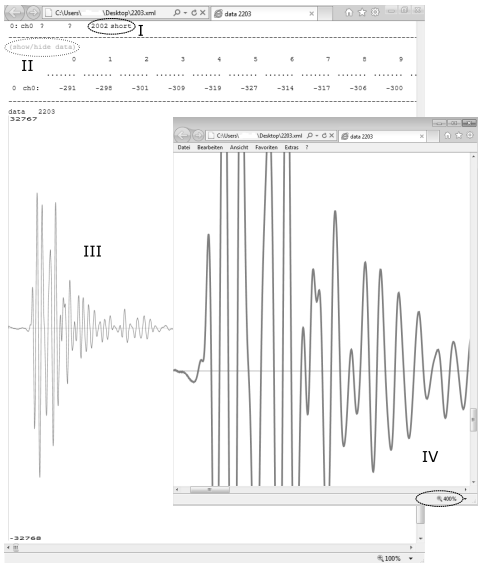

Figure 4: Visualization of an XML file in the web browser using xHTMl and embedded SVG. (I..data type of audio samples; II..audio signal as strings, can be hide; III..audio signal as SVG; IV..zoom in using browser zoom capability)

3.3. Editing

Manually editing an automatically generated speech recording annotation should be made by high quality annotation tools. But it is also possible to fix poor annotations while browsing the database by a web browser. All annotation marks inside a SVG are objects that can be removed or move along the time axis. It allows an elementary editing of the utterance content like changing word boundaries by moving the corresponding boundary mark. Using interactive SVG in combination with the web server made it possible to transmit the changes to the database. The changes are committed to web server, that will handle further processing like integration into a subversion system. For further editing it is possible to integrate java applets.

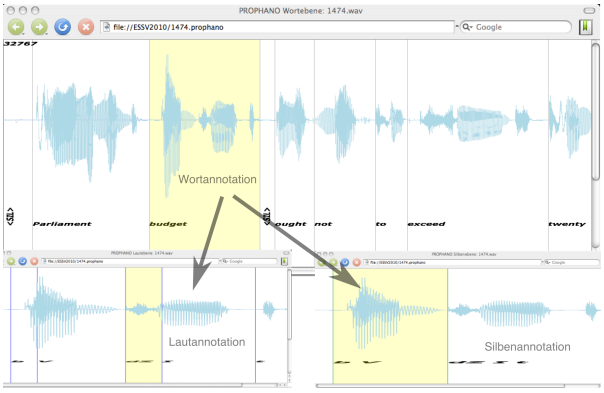

Figure 5: Visualization and editing of the annotation layer directly in the Browser with interactive SVG

4. Conclusions

The introduced speech processing framework creates phonetically and prosodically annotated speech databases. It uses structured XML files and a Document Type Definition (DTD) to ensure a valid data structure and interoperability between the used speech processing tools. All information layers of a utterance are collected in one data file that results in a very compact speech database. The visualization of the database is made by a normal web browser and the navigation through the different layers is like browsing a web page. If the user requests a utterance, the browser gets the requested file with all available information of the corresponding utterance. If a speech processing tool requests for the utterance at the underlying file server, the tool gets the same file as the browser. The uniform access is indispensable to get a consistent speech database.

5. References

[1] Wittenberg, S. and Jokisch, O. "Das Prosodisch-Phonetische Annotationssystem PROPHANO", In Proc. of 21st conference on Electronic Speech Signal Processing, pp. 71–76, Berlin, Germany, 2010.

[2] Jokisch O.; Wittenberg S.; Cuevas M.; Hussein H.; Strecha G.; Ding H.; Hoffmann R., "Towards an Automatic Process Chain for the Speech Corpora Annotation", Proc. SPECOM 2007, pp. 869-876, Moscow, October 2007.

[3] Wells J.C., "SAMPA computer readable phonetic alphabet", Handbook of Standards and Resources for Spoken Language Systems, Part IV, section B, Mouton de Gruyter, 1997.

[4] Kudrass, T. and Krumbein, T., "Rule-Based Generation of XML DTDs from UML Class Diagrams", Advances in Databases and Information Systems, vol. 2798, pp. 339–354, 2003.

[5] Dipper, S. and Goetze M., "Accessing Heterogeneous Linguistic Data Generic XML-based Representation and Flexible Visualization", In Proc. of 2nd Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pp. 206–210, Poznan, Poland, 2005.

[6] Milde, J.T. and Gut, U., "A Prosodic Corpus of Non-Native Speech", In Proc. of Speech Prosody 2002 Conference, pp. 503–506, 2002.

[7] Schroeder, M. and Trouvain, J., "The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching", International Journal of Speech Technology, 6, pp. 365-377,2003.