

# ***KielDat* – Data bank utilities for the *Kiel Corpus***

Matthias Pätzold

## **1 Overview**

*KielDat* represents the last processing stage in the Kiel system of labelling and segmenting speech signals. *KielDat* is a tool which facilitates working with the read and spontaneous speech corpora. It is designed so that phonetic analyses of differing complexity can be carried out.

*KielDat* is a collection of data base utilities for label files generated at the *ipds* Kiel. The user of *KielDat* should be familiar with the *ipds* labelling conventions laid out in Kohler, Pätzold, and Simpson (1995), Kohler, Lex, Pätzold, Scheffers, Simpson, and Thon (1994), and Kohler, Pätzold, and Simpson (1994).

## **2 Concept of the data base**

### **2.1 General idea**

The general idea was to design a data base that can be used to investigate a wide range of phonetic questions, taking into account that phoneticians have varying levels of computer knowledge. Commercial data base systems were not considered adequate for this task, since they store their data in unreadable formats and the user would have to use a programming language for data base queries. Furthermore, it would be far more difficult to convert the material according to the requirements of a special data base system. Because we are dealing with speech data and cannot foresee the complexity and types of questions to be asked, a more flexible data base system is required.

The data base generated using *KielDat* consists of a simple ASCII file, which has the advantage of being accessible with an editor or with operating system commands like *grep*. This ASCII file can also be processed with *awk*, which allows complex

queries using *awk* scripts.

## 2.2 Functions

The concept of *KielDat* does not only involve the generation of the data base itself, but also includes a comprehensive set of utilities to aid in the analysis of the data base. They allow the user to create a data base from a specified collection of label files and provide a highly flexible set of functions for working with the data base. The *KielDat* utilities are based on the *awk* programming language (see Aho et al. 1988, Close et al. 1993 and the *awk* manual page).

With some knowledge of UNIX and programming in *awk*, the user can utilize the powerful functions provided by *KielDat*. These functions allow a large number of data base queries to be performed by simply calling the appropriate function in the user's own script. The utility functions are organised as program libraries. The script which a user writes is an *awk* script with `#include` statements. For each library function used in the script the appropriate file must be included. In order to create an executable *awk* script the user's script is sent through a C-preprocessor. This concept enables the user to complete complex data base analysis while at the same time maintaining transparency in the scripts used.

The data base functions are intended to perform recurring tasks. They have been included in the data base to make it easier, especially for those who have little knowledge of computer programming, to write relatively complex data base queries with great efficiency regarding time, effort and reliability.

A typical example in which a relatively complex query is simplified using *KielDat* is the identification of labels marked as deleted, replaced or inserted. A user writing such checking functions from scratch has to have a comprehensive knowledge of the *ipds* labelling conventions as well as the ability to construct the necessary water-tight regular expressions in *awk*. The *KielDat* library, which includes such functions, avoids both of these problems, keeping the actual script written by the user compact and functionally transparent.

The following example shows the use of *KielDat* functions and at the same time illustrates how simple they are to use. The query prints all segments that are short vowels along with their points in time and their durations.

```
#include "lib/initDB"
#include "lib/isShortVowel"
#include "lib/isLabelDeleted"
#include "lib/extractLabelTimeDuration"

{
  nL = extractLabelTimeDuration(labels, times, durations)
  for (i=1; i<nL; i++) {
    if (isShortVowel(labels[i]) \
        && ! isLabelDeleted(labels[i]))
```

```

        print labels[i], times[i], durations[i]
    }
}

```

Beside the functions that deal with phonetic symbols there are further functions that are relevant for phonetic questions. These include mathematical functions, e.g. conversion from points in time to sample numbers, and statistical functions like the computation of arithmetic and geometric means and measures of variance.

The concept of data base functions make it possible to extend this important part of the data base by simply adding new functions and thus facilitate the work with the data base.

### 3 Make-up of a *KielDat* data base

Each line of a data base file represents a data base record. Each record comprises a number of fields separated by tabs. In certain cases tab-separated fields themselves are also further subdivided by other delimiting characters or character sequences. The name and content of tab-separated fields are presented in Table 1.

In general one record contains the information for one word. Word-external material, e.g., pauses, punctuation and the like, is treated differently. If only one such element occurs between two words it has its own record, if a sequence of such labels occurs between two words the sequence also only occupies a single record.

As was mentioned above a number of the tab-separated fields themselves may contain more than one sub-field. The sub-field separators are as follows:

- Single blank for ORTHO and ORTHOPOS
- Double blank for CANON

The VARIANT field is special as it always contains sub-fields separated by single blanks. Odd-numbered sub-fields in the VARIANT field are the labels, even-numbered sub-fields are double-figure indices used to access the time and duration fields.

If a stretch of speech is overlaid with articulatory (e.g. laughing) or non-articulatory noise (e.g. rustling of paper), this is excluded from the orthographic representation of the word in the ORTHO field. A representation of the word showing any overlay information is given in the OVERLAY field. The overlaying events are stored together with the word they are overlaying separated by \$. The overlaying elements themselves are separated from each other by @. If OVERLAY begins with <: then the word is the first item of the overlaid stretch, if OVERLAY ends with :> then the word is the last item of the overlaid stretch. The OVERLAYPOS is organised identically, containing the positions.

Due to the content of different parts of a label file, the ORTHO and CANON fields may be empty, whereas the VARIANT field is always filled. So, for instance, the sentence-initial marker #c: has no orthographic or canonical congeners.

Field	Symbolic Name	Description
1	ORTHO	orthographic representation
2	ORTHOPOS	position of word in orthographic part of the label file
3	OVERLAY	orthographic representation of an overlaying event
4	OVERLAYPOS	position of the overlaying event in orthographic part of the label file
5	CANON	canonical transcription
6	VARIANT	variant transcription
7	DATE	date and time of label file inclusion in data base
8	FILE	full filename of label file (including path)
9	BASENAME	basename of label file (i.e. no path or suffix)
10	SPEAKER	speaker index (includes turn number in dialogues, e.g. in the <i>Verbmobil</i> corpus)
11	GENDER	gender
12	LABELFORMAT	format of labels in either SAMPA or MIX notation <sup>a</sup>
13	SAMPLEFREQUENCY	sample frequency in Hertz
14	WORDBEGIN	start of word in seconds
15	WORDEND	end of word in seconds
16	TIME	(and all even-numbered fields following, i.e. 18, 20...) start times of labels
17	DURATION	(and all odd-numbered fields following, i.e. 19, 21...) durations of labels

---

<sup>a</sup>*KielDat* recognizes two formats, MIX and SAM. MIX is the label format used internally at *ipds*, SAM is the external format found on CD-ROMs#1–4 (IPDS 1994, 1995, 1996, 1997). Descriptions of the two file formats can be found in section 3.6.4 on p. 80.

Table 1: Data Base Fields

## 4 Utilities and functions

*KielDat* comprises:

- a small number of utilities used to create a data base and convert a user's script into an executable *awk* script;
- over 60 functions to access the data base.

The functions fall thematically into the following groups:

- Conversion, e.g. time to sample number; Hertz to mel; MIX to SAM label formats.

- Extraction, e.g. create an array of labels, times and durations of a word; extract the phonemic symbol from a label.
- Testing, e.g. is a label a vowel, a diphthong, nasalized?
- Removal, e.g. take off a label suffix; remove the truncation symbolization from a transcription.
- Mathematical, e.g. calculate arithmetic mean, median, standard deviation.
- Sort, e.g. sort a list of words.
- Miscellaneous, e.g. create a temporary filename; get the process id.

## 5 Working with a data base

Three steps are generally required to work with a data base:

1. generating the data base itself
2. writing a script using the utilities
3. converting this script to an executable *awk* script

Each of these steps are explained and exemplified in the following sections.

### 5.1 Generating the data base

The first step is to generate a data base from a selection of label files. The *ipds* label files must be divided into two sets, *PhonDat* and *Verbmobil*. The labelling conventions and file formats differ slightly. The *PhonDat* label files are those of the *Kiel Corpus of Read Speech* found on CD-ROM#1 (IPDS 1994), the *Verbmobil* label files are those of the *Kiel Corpus of Spontaneous Speech* on CD-ROM#2–4 (IPDS 1995, 1996, 1997).

However, although label files may differ in format, *data base* files created from either *PhonDat* or *Verbmobil* label files are formally identical. For each set of label files two programs have to be executed, *ph90kieldata1* and *ph90kieldata2* for *PhonDat* files, *vmkieldata1* and *vmkieldata2* for *Verbmobil* files.

### 5.2 Writing and running a script

Once a data base has been created from a selection of label files the analysis of the data base should be carried out using an *awk*-like script. This script should make use of a series of functions which have been specially tailored to manipulate data base records. The following should be observed when writing a script:

1. The first line must be

```
#include "lib/initDB"
```

in order to

- check whether the functions being used in a script are up-to-date
  - invoke *sysinit*
  - set the awk global variables FS (this should not be changed) and OFS to “\t”
  - set the global variables OFMT and CONVFMT to "%.10g".
2. Each data base function must be matched by an `#include` statement.
  3. Variable names should not contain an initial underscore or be in CAPITALS. The following global variables are set:
    - DBFS sub-field separator (set to " ")
    - STATVALUES[]
    - SYSVALUES[]
    - the symbolic names (see above) of the fields in a data base record.
  4. Comments can be in awk-style as long as # is not the first character of a line. It is perhaps most advisable to use C-style comments.

Having written a script this made into a ‘proper’ *awk*-script by sending it through C-preprocessor.

### 5.3 Examples

- **Example 1:** Print vowel labels and vowel durations of the female speakers who produced the Berlin sentences in the PhonDat corpus.

**Step 1** construct data base from all the Berlin sentences as they are found on CD-ROM#2, *Kiel Corpus of Read Speech Volume I*. This includes female and male speakers:

```
PROMPT> ph90kieldata1 -SAM /cd/ph90/berlin/k??/*.slh \
          | ph90kieldata2 > berlin.kdb
KielDat Generator Part I:
label convention:      PhonDat
file format:          SAM
sample frequency:     16000
processing             /cd/ph90/berlin/k01/k01be008.slh
```

*ph90kieldata1* preprocesses the SAM-formatted label files. Output is passed to *ph90kieldata2* which builds the actual data base. Output of *ph90kieldata2* is redirected to berlin.kdb.

**Step 2** write a script which extracts all female undeleted vowel labels from the data base:

```
#include "lib/initDB"
#include "lib/isRegularWord"
#include "lib/extractLabelTimeDuration"
#include "lib/isVowel"
#include "lib/math/round"
#include "lib/isLabelDeleted"

/* Don't forget to use C-style comments as this script
   will be sent through a C-preprocessor*/

isRegularWord($(ORTHO)) && $(GENDER) == "f" {

/* for each entry which is a regular word, extract the
   labels, their start times and durations */

    nl = extractLabelTimeDuration(labels, times, durations)

/* for each non-deleted vowel label, print the label
   and the duration in rounded millisecond values,
   followed by speaker index as a check */

    for(i = 1; i <= nl; i++)
        if(isVowel(labels[i]) && !isLabelDeleted(labels[i]))
            print labels[i], round(durations[i] * 1000), \
                $(SPEAKER)
    }
}
```

**Step 3** convert this script into an executable awk script.

```
PROMPT> kielfatmexe vowels.awk
trying to compose ./vowels.awk ---> ./vowels
PROMPT> chmod 744 vowels
```

(Note: The chmod command is only required the first time vowels is created)

**Step 4** run the script on the data base, redirecting the output to vowel.dat

```
PROMPT> vowel berlin.kdb > vowel.dat
```

- **Example 2:** make a sorted lexicon of regular lexical items together with their canonical and variant transcriptions from the label files on CD-ROM#3 *Kiel Corpus of Spontaneous Speech Volume II*.

**Step 1** construct data base:

```
PROMPT> vmkieldata1 -SAM /cd/unix/g*a/g*a/g*.slh \
          | vmkieldata2 > cd3.kdb
KielDat Generator Part I:
label convention:      Verbmobil
file format:          SAM
sample frequency:     16000
using file /usr/local/src/kieldata/KielDatGender
                    for gender information
processing /cd/unix/g10a/g101a/g101a007.slh
```

*vmkieldata1* preprocesses the SAM-formatted label files. Output is passed to *vmkieldata2* which builds the actual data base. Output of *vmkieldata2* is redirected to cd3.kdb.

**Step 2** write a script which extracts all regular lexical items and sorts them alphabetically. It is advisable to give this script a suffix such as “.awk”:

```
#include "lib/initDB"
#include "lib/cvToLexiconEntry"
#include "lib/isRegularWord"

/* Don't forget to use C-style comments as this script
   will be sent through a C-preprocessor */

isRegularWord($(ORTHO)) {
/* for each entry which is a regular word, convert
   orthographic representation canonical transcription
   and variant transcriptions to lexical entries */

    print cvToLexiconEntry($(ORTHO)), \
          cvToLexiconEntry($(CANON)), \
          cvToLexiconEntry($(VARIANT))
}
}
```

**Step 3** convert this script into an executable awk script.

```
PROMPT> kieldata2 lexicon.awk
trying to compose ./lexicon.awk ---> ./lexicon
PROMPT> chmod 744 lexicon
```

(Note: The `chmod` command is only required the first time lexicon is created)

**Step 4** run the script on the data base. Output is piped to UNIX `sort` command for initial dictionary sorting. This output is then piped to `uniq -c` removing multiple occurrences of the same variant entry and counting the number of identical entries. Output is redirected to `lexicon.dat`.

```
PROMPT> lexicon cd3.kdb | sort -d | uniq -c > lexi.dat
```

## 6 Summary

*KielDat* provides a set of utilities to create a structured data base from a selection of label files. The data base is a simple text file making it readily accessible with any number of editors and UNIX tools such as *grep*. For those who have some programming experience, *KielDat* also offers an extensive library of functions which enable complex analyses of the data base to be carried out. The library of functions is of course far from exhaustive, but it can easily be extended to include functions which arise out of new analyses.

A further feature of *KielDat* is that it is not restricted to work with the *Kiel Corpus* but can be applied to any set of label files which follow the *ipds* format and labelling conventions.

It is hoped that both *xassp* and *KielDat* will be made generally available in the near future providing a powerful processing package for use with the *Kiel Corpus* at both signal and symbolic levels of analysis.

## References

Aho, A. V., B. W. Kernighan, and P. J. Weinberger (1988). *The AWK Programming Language*. Addison-Wesley.

Close, D. B., A. D. Robbins, P. H. Rubin, and R. Stallman (1993). *The GAWK manual*. Free Software Foundation.

IPDS (1994). *The Kiel Corpus of Read Speech*, Volume 1, CD-ROM#1. Kiel: Institut für Phonetik und digitale Sprachverarbeitung.

IPDS (1995). *The Kiel Corpus of Spontaneous Speech*, Volume 1, CD-ROM#2. Kiel: Institut für Phonetik und digitale Sprachverarbeitung.

IPDS (1996). *The Kiel Corpus of Spontaneous Speech*, Volume 2, CD-ROM#3. Kiel: Institut für Phonetik und digitale Sprachverarbeitung.

IPDS (1997). *The Kiel Corpus of Spontaneous Speech*, Volume 3, CD-ROM#4. Kiel: Institut für Phonetik und digitale Sprachverarbeitung.

Kohler, K. J., G. Lex, M. Pätzold, M. T. M. Scheffers, A. P. Simpson, and W. Thon (1994). *Handbuch zur Datenaufnahme und Transliteration in TP14 von Verbmobil - 3.0*. Verbmobil Technisches Dokument Nr. 11.

Kohler, K. J., M. Pätzold, and A. P. Simpson (1994). *Handbuch zur Segmentation und Etikettierung von Spontansprache – 2.3*. Verbmobil Technisches Dokument Nr. 16.

Kohler, K. J., M. Pätzold, and A. P. Simpson (1995). *From scenario to segment: the controlled elicitation, transcription, segmentation and labelling of spontaneous speech*. AIPUK 29.