Munich AUtomatic Segmentation (MAUS)
Phonemic Segmentation and Labeling
using the MAUS Technique

F. Schiel
with contributions of
A. Kipp, Th. Kisler

Bavarian Archive for Speech Signals
Institute of Phonetics and Speech Processing
Ludwig-Maximilians-Universität München, Germany

www.bas.uni-muenchen.de
schiel@bas.uni-muenchen.de
Overview

- Statistical Segmentation and Labeling
- Super Short Introduction to MAUS
- Pronunciation Model : Building the Automaton
- Pronunciation Model : From Automaton to Markov Model
- Evaluation of Segmentation and Labeling
- MAUS Software Package
- MAUS Web Application
- MAUS Web Services
Let $\Psi$ be all possible Segmentation & Labeling (S&L) for a given utterance. Then the search for best S&L $\hat{K}$ is:

$$\hat{K} = \arg\max_{K \in \Psi} P(K|o) = \arg\max_{K \in \Psi} \frac{P(K)p(o|K)}{p(o)}$$

with $o$ the acoustic observation of the signal. Since $p(o) = \text{const}$ for all $K$ this simplifies to:

$$\hat{K} = \arg\max_{K \in \Psi} P(K)p(o|K)$$

with:

- $P(K) = \text{apriori probability for a label sequence}$,
- $p(o|K) = \text{the acoustical probability of } o \text{ given } K$ (often modeled by a concatenation of HMMs)
Statistical Segmentation and Labeling

S&L approaches differ in creating $\Psi$ and modeling $P(K)$

For example: *forced alignment*

$$||\psi|| = 1 \quad \text{and} \quad P(K) = 1$$

hence only $p(o|K)$ is maximized.

Other ways to model $\Psi$ and $P(K)$:

- phonological rules resulting in $M$ variants with $P(K) = \frac{1}{M}$
- phonotactic n-grams
- lexicon of pronunciation variants
- **Markov process** (MAUS)
Short Introduction to MAUS

![Diagram of MAUS process]

1. **TEXT**
2. **PHONOLEX**
3. **PHONRUL**
4. **REFRUL**

- **Lexicon Lookup**
- **Generator**
- **Viterbi**
- **Retinacement**

Result: **MAUS Output**
Short Introduction to MAUS
Start with the orthographic transcript:

*heute Abend*

By applying lexicon-lookup and/or a test-to-phoneme algorithm produce a (more or less standardized) citation form in SAM-PA:

hOYt@ ?a:b@nt

Add word boundary symbols #, form a linear automaton $G_c$:

![Diagram of a linear automaton](image)
Extend automaton $G_c$ by applying a set of substitution rules $q_k$ where each $q_k = (a, b, l, r)$ with:

- $a$: pattern string
- $b$: replacement string
- $l$: left context string
- $r$: right context string

For example, the rules

$(/ @n/, /m/, /b/, /t)$ and $(/ b@n/, /m/, /a:/, /t/)$

generate the reduced/assimilated pronunciation forms

$/?a:bmt/$ and $/?a:mt/$

from the canonical pronunciation

$/?a:b@nt/ (evening)$
Applying the two rules to $G_c$ results in the automaton:
From Automaton to Markov Process

Add transition probabilities to the arcs of $G(N, A)$

- Case 1: all paths through $G(N, A)$ are of equal probability
  Not trivial since paths can have different lengths!
  Transition probability from node $d_i$ to node $d_j$:

$$P(d_j|d_i) = \frac{P(d_j)N(d_i)}{P(d_i)N(d_j)}$$

$N(d_i)$: number of paths ending in node $d_i$
$P(d_i)$: probability that node $d_i$ is part of a path

$N(d_i)$ and $P(d_i)$ can be calculated recursively through $G(N, A)$ (see Kipp, 1998 for details).
Example:
Markov process with 4 possible paths of different length

Total probabilities:
\[
1 \cdot \frac{3}{4} \cdot \frac{1}{3} \cdot 1 \cdot 1 = \frac{1}{4} \\
1 \cdot \frac{1}{4} \cdot 1 \cdot 1 = \frac{1}{4} \\
1 \cdot \frac{3}{4} \cdot \frac{1}{3} \cdot 1 = \frac{1}{4} \\
1 \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot 1 \cdot 1 = \frac{1}{4}
\]
Case 2: all paths through $G(N, A)$ have a probability according to the individual rule probabilities along the path through $G(N, A)$

Again not trivial, since contexts of different rule applications may overlap!
This may cause total branching probabilities > 1

*Please refer to Kipp, 1998 for details to calculate correct transition probabilities.*
True HMM: add emission probabilities to nodes $N$ of $G_c$.

$\Rightarrow$ Replace the phonemic symbols in $N$ by mono-phone HMM.

The search lattice for previous example:
Word boundary nodes ‘#’ are replaced by an optional silence model:

Possible silence intervals between words can be modeled.
Evaluation of Segmentation and Labeling

How to evaluate a S&L system?

Required: reference corpus with hand-crafted S&L (‘gold standard’).

Usually two steps:

1. Evaluate the accuracy of the label sequence (transcript)
2. Evaluate the accuracy of segment boundaries
Evaluation of Label Sequence

Often used for label sequence evaluation: Cohen’s $\kappa$

$\kappa = \text{amount of overlap between two transcripts (system vs. gold standard)}$; independent of the symbol set size (Cohen 1960).

We consider $\kappa$ not appropriate for S&L evaluation, since
- no gold standard exists in phonemic S&L
- different symbol set sizes do not matter in S&L
- the task difficulty is not considered (e.g. read vs. spontaneous speech)
Evaluation of Label Sequence

Proposal: *Relative Symmetric Accuracy (RSA) =*

= the ratio from average symmetric system-to-labeler agreement $\hat{SA}_{hs}$ to average inter-labeler agreement $\hat{SA}_{hh}$.

$$RSA = \frac{\hat{SA}_{hs}}{\hat{SA}_{hh}} \times 100\%$$
Evaluation of Label Sequence

German MAUS:
- 3 human labelers
- spontaneous speech (Verbmbil)
- 9587 phonemic segments

Average system - labeler agreement
\[ \hat{SA}_{hs} = 81.85\% \]

Average inter - labeler agreement
\[ \hat{SA}_{hh} = 84.01\% \]

Relative symmetric accuracy
\[ RSA = 97.43\% \]
Evaluation of Segmentation

- No standardized methodology
- Problem: insertions and deletions
- Solution: compare only matching segments
- Often: count boundary deviations greater than threshold (e.g. 20msec) as errors
- Better: deviation histogram measured against all human segmenters
German MAUS:

Note: center shift typical for HMM alignment
MAUS software package:

ftp://ftp.bas.uni-muenchen.de/pub/BAS/SOFTW/MAUS

MAUS package consists of

- basis script maus
- corpus processor maus.corpus
- adaptive maus maus.iter
- chunk segmentation processor maus.trn
- helper programs: visualization, graph generator etc.
- parameter sets for supported languages
- test benchmarks
MAUS installation requires:

- UNIX System V or cygwin
- Gnu C compiler
- HTK (*University of Cambridge*)
- awk, sox

Current language support:

deu, eng, ita, aus (with pronunciation modelling)
hun, ekk, por, spa, nld, sampa (without modelling)

```bash
maus BPF=file.par \ SIGNAL=file.wav LANGUAGE=eng \ OUT=file.TextGrid OUTFORMAT=TextGrid
```
How to adapt MAUS to a new language?

Several possible ways (in ascending performance and effort):

- Use SAM-PA ‘language’ (collective MAUS phoneme set). No pronunciation modelling possible.
  
  **Effort:** nil
  
  **Performance:** for some languages surprisingly good.
Hand craft pronunciation rules (depending on language not more than 10-20) and run MAUS in the ’manual rule set’ mode.

*Effort:* small

*Performance:* Very much dependent of the language, the type of speech, the speakers etc.

Adapt HMM to a corpus of the new language using an iterative training schema (*script* `maus.iter`). Corpus does not need to be annotated.

*Effort:* moderate (if corpus is available)

*Performance:* For most languages very good, depending on the adaptation corpus (size, quality, match to target language etc.)
Retrieve statistically weighted pronunciation rules from a corpus. The corpus needs to be at least of 1 hour length and segmented/labeled manually.

*Effort:* high.

*Performance:* Unknown.
MAUS Web Interface

http://clarin.phonetik.uni-muenchen.de/BASWebServices/

- **WebMAUS**: web interface to the latest version of MAUS
- **Pros:**
  - no local installation necessary
  - runs on all platforms (even SmartPhones)
  - text-normalization and text-to-phoneme (partially)
- **Cons:**
  - no adaptation to new languages
  - no application of proprietary rule sets
  - no iterative adaptation mode
WebMAUS Basic: \textit{Signal + Text} $\rightarrow$ \textit{Segmentation}

- simple, robust
- includes text-normalisation, tokenization and text-to-phoneme conversion
- no control of parameters or input (except language)
- supported languages: deu, hun, eng, nld, ita
- supported output: TextGrid (praat)
WebMAUS General

WebMAUS General: *Signal + Phonology* -> *Segmentation*

- full control of all MAUS options
- phonologic input allows fine tuning
- requires input in BAS Partitur Format (BPF)
- supported output BPF, TextGrid, Emu
- supported languages:
  - deu, eng, ita, aus, hun, ekk, por, spa, nld, sampa
WebMAUS Multiple

WebMAUS Multiple: *Signals + Texts* -> *Segmentations*

- drag & drop of input files
- features like WebMAUS Basic
- batch processing of unlimited file pairs
web service = direct call to a server

MAUS web services can be used within programming languages or scripts or from the command line, e.g.:

curl -v -X POST -H 'content-type: multipart/form-data' \ -F LANGUAGE=deu -F TEXT=@file.txt -F SIGNAL=@file.wav \ http://clarin.phonetik.uni-muenchen.de/ \ BASWebServices/services/runMAUSBasic

To get started call:

curl -X GET \ http://clarin.phonetik.uni-muenchen.de/BASWebServices/services/help
The script `maus.web` (in MAUS package) can be used like the original `maus` script, but issues web service calls.

```
maus.web BPF=file.par \ 
SIGNAL=file.wav LANGUAGE=eng \ 
OUT=file.TextGrid OUTFORMAT=TextGrid
```
References

- MAUS: ftp://ftp.bas.uni-muenchen.de/pub/BAS/SOFTW/MAUS
- CLARIN: http://www.clarin.eu/
Questions?