11 Münchner AUtomatische Segmentierung – MAUS

11.1 Grundprinzip der automatischen Segmentierung

Oft braucht ein Phonetiker für eine Sprachaufnahme eine Wort- oder Phonem-Segmentierung. Der normale Weg ist, diese Segmentierung von einem geschulten Phonetiker mit Hilfe eines Segmentier- und Etikettierprogramms (z.B. `praat`, `WaveSurfer`) anfertigen zu lassen. Je nach Aufgabe benötigt man dafür ein mehrfaches der Echtzeit (= die Dauer des Sprachsignals).

- Einfache orthographische Verschriftung: 25fache Echtzeit
- Verschriftung mit Tagging (z.B. `Verbmobil`): 80fache Echtzeit
- phonologisches Transkript: 50fache Echtzeit
- phonetische Segmentierung und Etikettierung (S&L) in SAM-PA: 120-180fache Echtzeit
- phonetische S&L in IPA: bis zu 250fache Echtzeit

Um den Vorgang zu beschleunigen, kann man automatisch segmentieren: ausgehend von einem vorhandenen phonetischen Transkript oder einer orthographischen Verschriftung wird ein Modell der Aussprache gebildet und dieses mit Hilfe von Hidden Markov Modellen auf das Signal abgebildet[1]

Oft wird als 'Modell' einfach die Sequenz der Phoneme in einer Standard-Aussprache (Aussprache-Lexikon) des jeweiligen Satzes verwendet, also z.B.

> Orthographie: Heute haben wir schönes Frühlingswetter
> Transkript: hOYt@ ha:b@n vI6 S2:n@s fRy:lINsvEt6[2]

Mit dem *Viterbi*-Algorithmus wird diese Sequenz auf das Signal abgebildet, was dann eine S&L ergibt, die genau diese vorgegebenen Phoneme enthält (Viterbi alignment).

Problematisch ist dies, wenn die tatsächlich realisierte Sprache nicht mit der Standardaussprache (kanonischen Aussprache) übereinstimmt, wenn z.B. statt /ha:b@n vI6/ nur ein /ham v6/ geäußert wurde. Das kommt sogar in gelesener Sprache ständig vor. MAUS versucht, auch solche Fälle mit zu berücksichtigen, indem es für jeden zu segmentierenden Satz ein komplizierteres Modell mit vielen, statistisch gewichteten Aussprachealternativen berechnet (Hypothesenmodell, s. Abb. 11.1), dieses dann mit Hilfe des Viterbi-Algorithmus auf das Sprachsignal abbildet und somit automatisch die plausibelste Aussprache-Segmentierung findet. Das Hypothesenmodell wird mit Hilfe von maschinell gelernten, statistischen Aussprachevarianten gebildet, welche aus einem großen Korpus (ca. 1h Sprache) gelernt wurden.

Alternativ kann MAUS auch klassische phonologische Regeln für alternative Aussprachen verwenden. Diese müssen aber von Hand für den jeweiligen Fall kodiert werden.

MAUS segmentiert in das phonologisch orientierte SAM Alphabet mit 44 Phonem-Klassen (Deutsch). Die Genauigkeit ist nicht so gut, wie es ein menschlicher Segmentierer erreicht (ca. 96% davon). MAUS gibt es für Deutsch, Englisch, Ungarisch und Italienisch, und kann relativ leicht an weitere Sprachen angepasst werden (z.B. war für Isländisch gar keine Anpassung notwendig).

Im Folgenden wollen wir Daten aus der parallelen Lehrveranstaltung *Akustische Segmentierung* (Felicitas Kleber) mit MAUS segmentieren und uns die Ergebnisse mit `praat` anschauen. Wenn weder Sie noch Ihr Praktikumspartner an dieser Veranstaltung teilnehmen, kopieren Sie bitte die Daten von einem Kommilitonen.

[1] Eine automatische Segmentierung mit automatischer Spracherkennung, d.h. ohne jede Information, was in dem Sprachsignal gesprochen wurde, liefert nur sehr fehlerhafte Ergebnisse.

[2] Phonetische Symbole in SAM-PA.

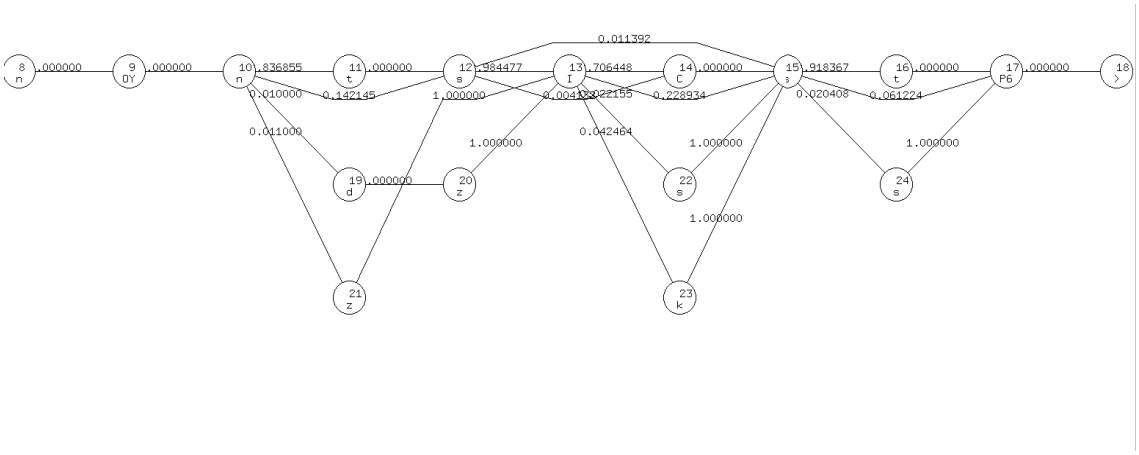

Abbildung 1: Aussprachemodell des Wortes 'neunzigster'

11.2 Das MAUS Programm

MAUS besteht aus einen Paket von Skripten und Programmen. Das wichtigste Kommando für uns ist `maus`. Rufen Sie es auf und überfliegen Sie die lange Hilfe-Ausgabe:

```
cip1 % maus | less
```

MAUS kann entweder eine kanonische Aussprache (citation form) von der Kommandozeile lesen:

```
cip1 % maus OUT=Result.mau SIGNAL=signal.nis KANSTR="f i: r # Q U n t # t s v a n t s I C"
```

(hier nur drei Wörter 'vier und zwanzig') oder es liest die Aussprache aus der Spur 'KAN' eines BPF Files:

```
cip1 % maus OUT=Result.mau SIGNAL=signal.nis BPF=signal.par
```

In `signal.par` sollte dann z.B. folgendes stehen[3]:

```
LHD: Partitur 1.2
REP: Munich, Germany
SNB: 2
SAM: 22050
SBF: 01
SSB: 16
NCH: 1
SPN: 3348
LBD:
ORT:  0  vier
ORT:  1  und
ORT:  2  zwanzig
KAN:  0  fi:r
KAN:  1  QUnt
KAN:  2  tsvantsIC
```

[3]Die Spur 'ORT' kann auch weggelassen werden, wenn eine Spur 'KAN' vorhanden ist.

Hat man keine kanonische Aussprache-Spur zur Verfügung, liest MAUS die orthographische Spur 'ORT' und berechnet daraus automatisch eine möglichst gute kanonische Aussprache.

Das Ergebnis der S&L wird in der Datei `Result.mau` in Form einer 'MAU' Spur (phonetische Segmentierung) gespeichert:

```
MAU:  0   3425  767    f
MAU:  0   4193  1289   i:
MAU:  0   5483  989    6
MAU:  -1  6473  319    <p:>
MAU:  1   6793  856    U
MAU:  1   7650  900    n
MAU:  1   8550  456    t
...
```

MAUS hat hier automatisch das /R/ in 'vier' vokalisiert, den glottal Stop vor 'und' weggelassen und eine kurze Pause zwischen 'vier und 'und' segmentiert.

Alternativ kann MAUS das Ergebnis auch in einem `praat` TextGrid File speichern:

```
cip1 % maus OUT=Result.TextGrid OUTFORMAT=TextGrid SIGNAL=signal.nis BPF=signal.par
```

11.3 Übung mit MAUS

1. Wählen Sie 10 von Ihnen aufgenommene Sätze aus der Veranstaltung 'Akustische Segmentierung' und kopieren Sie die Sound Files in Ihr Arbeitsverzeichnis. Nennen Sie diese `Satz_##.wav`, wobei `##` von 01 bis 10 geht.

2. Starten Sie einen Editor und schreiben Sie die gesprochenen Wörter jeder Aufnahme in jeweils eine Datei mit dem Namen `Satz_##.txt`. In jeder dieser Textdateien steht also genau eine Zeile mit dem Text der Aufnahme.

3. Schreiben Sie ein Skript `make_bpf`, welches alle 10 Textdateien nacheinander liest und zu jeder eine BPF Datei `Satz_##.par` erzeugt, welche die gesprochenen Wörter in der Spur 'ORT' enthält. Ein solches BPF File muss dann so aussehen:

   ```
   LHD: Partitur 1.2
   SNB: 2
   SAM: <hier steht die Abtastrate>
   SBF: 01
   SSB: 16
   NCH: 1
   LBD:
   ORT:  0  wort1
   ORT:  1  wort2
   ORT:  2  wort3
   ...
   ```

4. Testen Sie zunächst mit der ersten Aufnahme, ob MAUS damit funktioniert:

   ```
   cip1 % maus OUT=Satz_01.TextGrid OUTFORMAT=TextGrid SIGNAL=Satz_01.wav BPF=Satz_01.par
   ```

Wenn Sie eine Fehlermeldung bekommen, versuchen Sie zu ergründen, wo der Fehler liegt.

Wenn Sie keine Fehlermeldung bekommen, schauen Sie sich mit `praat` das Signal und die erzeugte Segmentierung an. Ist das alles korrekt?

5. Schreiben Sie nun ein Skript `make_maus`, welches alle 10 Aufnahmen nacheinander segmentiert. Kontrollieren Sie die Ergebnisse.