# Kapitel 2

# HTK–Praktikum

## 2.1 Allgemeines

Es soll ein einfacher Spracherkenner für spontan gesprochenes Deutsch entworfen und implementiert werden.

Für die praktischen Übungen am Rechner: Jeder Teilnehmer wird einer Gruppe 1 - 6 zugeordnet (GROUP# bedeutet zukünftig die Gruppe des Teilnehmers). Die Teilnehmer einer Gruppe sollten zusammen an einem Rechner arbeiten.

Einloggen:
User: htk
Password: (wird bekannt gegeben)

```
% cd HTK-KURS/GROUP#
```

BITTE NUR IM ZUGEWIESENEN GRUPPEN-VERZEICHNIS ARBEITEN!

## 2.2 Vorbereitung der notwendigen Daten

Zunächst müssen die notwendigen Resourcen zur Verwendung in HTK aufbereitet werden.

### 2.2.1 Sprach Korpora

Die verwendeten Sprach-Korpora werden sinnvollerweise nur einmal in Folgen von Merkmalsvektoren vorverarbeitet, da dieser Schritt rechenaufwendig ist und jedesmal durchgeführt werden muß, wenn auf Sprachdaten zugegriffen wird. Außerdem sind vorverarbeitete Sprachdaten weniger speicherintensiv als das reine Sprachsignal (Daten-Kompression).

Wir brauchen (im Prinzip) 3 Korpora: Training, Development und Test Die Methode ist für alle drei exakt dieselbe:

- Erstellen eines 'Skripts' mit einer Liste der zu verarbeitenden Sprachfiles und der resultierenden vorverarbeiteten HTK files (zweispaltige Liste)

- Definition der gewünschten Merkmale in einem config file
- Durchführen der Vorverarbeitung mit dem Befehl

```
% HCopy -C config-file -S script-file
```

Schauen Sie sich das hier verwendete config-file an:

```
% less ../PRECONFIG
```

Die wichtigsten Einträge sind wie folgt: Die Quelle (SOURCEKIND) wird als Sprachsignal (WAVEFORM) definiert; das File-Format ist NIST (SOURCEFORMAT); der Abstand zweier Abtastwerte (SOURCERATE) ist 625 * 100 Nanosekunden (alle Zeitangaben in HTK erfolgen in Vielfachen von 100 ns). Welcher Abtastrate in Hz entspricht dies?

Das Ausgabe-Format (TARGETFORMAT) ist HTK; die resultierenden htk Files enthalten Merkmalsvektor mit der Struktur (TARGETKIND):

MFCC_E_D_A : Mel-Frequency-Ceptral-Koeffizienten (MFCC), logarithmierte Energy (E), 1. Ableitung (D), 2. Ableitung (A)

Zu Mel-Frequency-Cepstrum siehe Kap. 5.4 in [2]. Das Cepstrum wird mit Filterordnung 12 berechnet (NUMCEPS); der Abstand zweier Vektoren ist 10 msec (TARGETRATE) und die Breite des Analysewindows (Hamming) ist 20 msec (WINDOWSIZE).

Welche Dimensionalität haben also die Merkmalsvektoren?

WARNUNG: Da es sich um beträchtliche Datenmengen handeln kann, muß vorher sorgfältig überlegt werden, wo die resultierenden htk Files abgelegt werden, damit es nicht zu einem Plattenüberlauf kommt.

Um Zeit zu sparen, wurde dieser (einfachste) Schritt bereits durchgeführt. Die Daten liegen unter:

/data/data10_2/HTK-VM/<dialog>/<htk-file>

(Bemerkung: Trainings-, Test- und Development-Set sind zwar an der gleichen Stelle gespeichert, sind aber trotzdem disjunkt!)

In HTK formatierte Dateien sind normalerweise binär kodiert, d.h. sie sind nicht mit einem normalen Text-Editor zu bearbeiten. Zur Kontrolle der Inhalte von HTK-Files dient das Kommando 'HList'.

```
% cd /data/data10_2/HTK-VM/m018n/
% HList -h -o m018n000.htk | less
```

Handelt es sich um ein HTK-File mit Merkmalsvektoren, werden diese mit '0' beginnend durchnummeriert ausgegeben. Nach jedem 10. Element wird automatisch ein Zeilenvorschub gemacht. Mit verschiedenen Optionen kann zusätzliche Information ausgegeben werden, z.B. die Struktur der gespeicherten Merkmalsvektoren.

```
---------------------------------------------------------------
SIGNALISIEREN SIE DAS ENDE DER UEBUNG - BESPRECHUNG VON FRAGEN
---------------------------------------------------------------
```

### 2.2.2 Skript-Files

Ein 'Skript-File' ist ganz simpel eine Liste mit Dateinamen von Dateien, die von einem HTK-Tool verarbeitet werden sollen. Der Name 'Skript' ist einigermaßen missverständlich: es handelt sich nicht um ein Skript im Sinne von Shell-Programmierung! Da aber dieser Begriff in [2] durchgängig so verwendet wird, haben wir ihn hier beibehalten. Normalerweise steht ein Filename pro Zeile; bei HTK-Tools, die Input und Output files haben, stehen diese hintereinander in einer Zeile (nur bei 'HCopy').

Entsprechende 'Skripte' für unsere Korpora befinden sich in:

- ~/HTK-KURS/TRAIN.slist (Training-Sets)
- ~/HTK-KURS/DEV.slist (Development-Set)
- ~/HTK-KURS/TEST.slist (Test-Set)

Schauen Sie sich eines dieser Files mit 'less' an.

Im Gegensatz zu den Skript-Files für die Vorverarbeitung steht hier nur jeweils ein Inputfile in einer Zeile (Outputfiles sind nicht nötig, da die meisten HTK-Tools die Namen der Output-Files automatisch generieren; einzige Ausnahme ist HCopy).

Fragen:
Wieviele Aufnahmen enthält das DEV Testset?
Wieviele Aufnahmen enthält das *gesamte* Trainingsset?

### 2.2.3 Master Label Files (MLFs)

Zusätzlich zu den eigentlichen Signalen braucht man ständig sogenannte symbolische Informationen zu den Sprachfiles. Ein einfaches Beispiel dafür ist die gesprochene Wortfolge in der Äußerung; ein komplizierteres Beispiel wären phonetische Segmentierungen. Auch die Erkennungsergebnisse von HTK werden als symbolische Information gespeichert. Man nennt solche Files auch 'Label-Files'.

HTK kann entweder einzelne Files verwalten (pro Sprachsignal-File ein oder mehrere Label-Files) oder - wesentlich effizienter - diese in sog. Master Label Files für einen ganzen Korpus zusammenfassen. Auch MLFs sind einfache Text-Dateien, d.h. sie sind mit einen normalen Editor lesbar und änderbar. Ein MLF besteht aus einer 'magic number' in der ersten Zeile und dann aus einer Kette von Labelfiles, die durch einen einzelnen '.' in einer Zeile und einen definierten String mit dem Filenamen des nächsten Label-Files getrennt werden. Die Reihenfolge spielt keine Rolle: wenn ein HTK-Tool zu einem bestimmten Sprachsignalfile namens g456ac.htk ein Labelfile sucht, richtet es sich nur nach dem Filenamen.

Wir brauchen für unseren Erkenner:

- Wort-Referenzen für Training-, Development- und Test-Set
- Segment-Referenzen für Training-Set

Für jeden Turn im Verbmobil-Korpus existiert ein sog. Partitur-File, welches unter anderem die Liste der gesprochenen Wörter (einschl. Hesitationen) enthält (ORT-Spur) und eine automatisch erzeugte phonetische Segmentierung (MAU-Spur). (Eine detailierte Beschreibung des Partitur-Formats ist unter `http://www.phonetik.uni-muenchen.de/Bas/BasFormatsdeu.html` zu finden.) Die Partitur-Files für beide Sets sind unter ~/HTK-KURS/TRAIN.Partitur, DEV.Partitur und TEST.Partitur abgespeichert.

Schauen Sie sich mit 'less' ein solches Partitur-File an. Finden Sie die 'ORT'- und die 'MAU'-Spur.

Aus diesen Spuren sollen nun die erforderlichen MLF erzeugt werden.

### Wort-Referenzen

Der einfachste Weg zur Erstellung eines MLFs ist der Befehl HLEd. HLEd ist ein skript-gesteuerter Spezial-Editor für Label-Files. Der allgemeine Aufruf ist:

```
% HLEd [Optionen] Cmdfile Labelfiles ...
```

Das Cmdfile ist ein Text-File, welches die auszuführenden HLEd-Kommandos enthält. HLEd dient primär zur Manipulation von Label-Files oder MLF, z.B. können Labels vertauscht, gelöscht, eingefügt, konvertiert, sortiert, expandiert (z.B. zu Triphones) werden.

HLEd braucht allerdings als Input files pro Turn einfache Label-Files mit genau einem Wort pro Zeile, z.B.

```
guten
Tag
wir
m"u"sten
noch
einen
...
```

Wir erzeugen also zunächst aus jedem Partitur-File ein Label-File in dieser Art:

```
% cd ~/HTK-KURS/GROUP#
% mkdir TEST.Label
% cd TEST.Label
% foreach pf ( ~/HTK-KURS/TEST.Partitur/*.par )
   set desti = ${pf:t:r}.lab
   cat $pf | gawk -v INPUT=$pf -f ~/HTK-KURS/par2lab.awk | \
     sed 's/^"/\\"/' >! $desti
   echo $desti
 end
% ls
% less m123nxx0_003_ANK.lab
```

Das AWK-Skript 'par2lab.awk' extrahiert nicht nur die puren Wörter, sondern zusätzlich noch die 'Wörter' !SIL (für Pausen), !BREATH (für Atmen), !LAUGH für Lachen, !ANOISE für alle übrigen artikulatorischen Geräusche (z.B. Husten) und !NOISE (für alle sonstigen Geräusche). Außerdem fügt es an jedem Satzbeginn und -ende ein zusätzliches Pausen-Label ein, weil satzinitiale und -finale Pausen in der Transliteration nicht extra markiert wurden, aber häufig vorkommen.

Das SED-Skript sorgt dafür, daß Wörter mit einem `'"'` zu Beginn (z.B. `'"uberlegen'`) mit einem Backslash quotiert werden (sonst würde HLEd erwarten, daß es sich um ein quotiertes Wort handelt!)

Nun laden wir diese Labelfiles in ein MLF, aber so, daß der Pfad 'offen' bleibt, d.h. nur der Filename soll enthalten sein, aber kein absoluter Pfad.

```
% cd ~/HTK-KURS/GROUP#
% touch dummy.led
% cd TEST.Label
% HLEd -T 1 -l '*' -i ../TEST.REF.mlf ../dummy.led  *.lab
% cd ..
% less TEST.REF.mlf
```

Um Zeit zu sparen, habe ich die Label-Files für das Trainings- und Development-Set bereits erzeugt. Sie sind in den Files TRAIN.REF.mlf und DEV.REF.mlf im Verzeichnis HTK-KURS gespeichert. Kopieren Sie diese in Ihr Gruppenverzeichnis.

Beachte, daß HLEd auch schon dafür gesorgt hat, daß Wörter mit Sonderzeichen (z.B. können) quotiert wurden.

```
-----------------------------------------------------------------
SIGNALISIEREN SIE DAS ENDE DER UEBUNG - BESPRECHUNG VON FRAGEN
-----------------------------------------------------------------
```

**Segmentale MLF**

Für den Bootstrap unserer Modelle brauchen wir ein MLF mit phonetischen Segmentierungen für zumindest einen Teil des Trainingskorpus (ideal wäre der ganze Korpus, aber der steht meistens nicht zur Verfügung)

In unserem Beispiel wollen wir den Teil 1 des Trainingskorpus als Bootstrap-Daten verwenden. Der Einfachheit halber verwenden wir hier die MAU-Spur der Partitur-Files.

In den MAU-Spuren ist die tatsächliche Aussprache (nicht die kanonische) der gesprochenen Sätze im SAM-PA verzeichnet. SAM-PA ist ein computerlesbares phonetisches Alphabet, das sehr viel gröber ist als z.B. IPA. Wenn Sie noch nie mit SAM-PA gearbeitet haben, schauen Sie sich in Ihrem Webbrowser folgende Page an: `http://www.phonetik.uni-muenchen.de/Bas/BasSAMPA`

Analog zu Punkt a) erzeugen wir also jetzt ein Segmentfile mit genau einen Segment pro Zeile. Allerdings brauchen wir jetzt außer der Aussprache noch die Zeitinformation von wo bis wo das phonetische Segment im Zeitsignal liegt. Diese Zeitangaben bilden die beiden ersten Spalten des Segmentfiles, die dritte Spalte enthält das Label selber. Die Zeitangaben sind in Einheiten von 100 nsec anzugeben ( 100 nsec = 0.0000001 sec ). Außerdem müssen wir die SAM-PA-Label /2:/, /6/ und /9/ umbenennen in /_2:/, /_6/ und /_9/, weil HTK keine führenden Zahlen in Modellnamen erlaubt. Dann gibt es in der MAU-Spur noch die speziellen Segmente /<p:>/ (Pause), /<nib>/ (nichtsprachlicher Bereich) und /<usb>/ (unspezifiziert, z.B. Unverständliches). Letzteres wird auf /<nib>/ abgebildet, da es sehr selten vorkommt. Schließlich wird noch der stimmhafte Esch-Laut /Z/ ('Garage') auf /S/ abgebildet, weil er zu selten vorkommt, als daß man ihn modellieren könnte.

```
% cd ~/HTK-KURS/GROUP#
% mkdir TRAIN.Segments
% cd TRAIN.Segments
% foreach htk ( `cat ~/HTK-KURS/TRAIN.slist` ) # nach links ge-kippte Hochkommata!
   set bs = ${htk:t:r}
   set pf = ~/HTK-KURS/TRAIN.Partitur/${bs}.par
   cat $pf | grep '^MAU:' | gawk -f ../../mau2lab.awk > ${bs}.lab
   echo $bs
```

```
   end
% ls
% less m123dxx0_003_KLA.lab
```

Nun laden wir diese Labelfiles in ein MLF namens `TRAIN.mlf`, aber so, daß der Pfad 'offen' bleibt, d.h. nur der Filename soll enthalten sein, aber kein absoluter Pfad.

```
FINDEN SIE SELBST HERAUS, WIE DER BEFEHL HIER HEISSEN MUSS
% cd ..
% less TRAIN.mlf
```

Schauen Sie sich das AWK-Skript ∼/HTK-KURS/mau2lab.awk an und versuchen Sie zu verstehen, wie das Skript arbeitet. Überprüfen Sie, ob alles richtig erledigt wurde. Stellen Sie fest, wie viele Segmente das MLF enthält (ungefähr). Warum war die Abbildung des /Z/ auf /S/ notwendig?

TYPISCHE FEHLER, DIE HIER PASSIEREN KOENNEN:

- Wenn zuviele Label-Files vorhanden sind, meldet sich die Shell mit der Fehlermeldung: 'Arg list too long'
  Lösung: Man erstellt mehrere MLFs, die jeweils nur eine Untergruppe der Label-Files enthalten, z.B. alle Dialoge, die mit 'm' beginnen:

  ```
  % HLEd -l '*' -i ../TRAIN.m.mlf ../dummy.led m*.lab
  ```

  Danach fasst man die verschiedenen MLFs mit 'cat' zusammen und löscht mit einem Editor die überflüssigen Zeilen mit 'magic numbers' darin.

- Man hat ein MLF erzeugt, aber andere HTK-Tools behaupten, daß sie darin die richtigen Label-Files nicht finden können. Lösung: Man hat wahrscheinlich die Option '-l *' vergessen. Diese Option bewirkt, daß in den MLFs die absoluten Pfade der Label-Files durch '*' ersetzt werden (Sonst verlangen alle HTK-Tools, daß der im MLF verzeichnete Pfad genau mit dem Pfad des Signal-Files übereinstimmt!). Der Joker '*' sorgt dafür, daß die Label-Files unabhängig vom Pfad der Signal-Files immer gefunden werden (vorausgesetzt der Name ist auch wirklich eindeutig!)

```
----------------------------------------------------------------
SIGNALISIEREN SIE DAS ENDE DER UEBUNG - BESPRECHUNG VON FRAGEN
----------------------------------------------------------------
```

### 2.2.4 Dictionary

Das Dictionary wird für den Test, aber ev. auch für das Training (Stichwort 'embedded training') benötigt. Die Größe des Dictionaries beeinflusst ganz erheblich die Erkennungsleistung und die Erkennungszeit. Ganz streng genommen müsste man das Dictionary allein aus dem vorhandenen Trainings- Material erstellen. Das hat zur Folge, daß beim Test der Fall eintreten kann, daß ein unbekanntes Wort gesprochen wird, das der Erkenner auf keinen Fall erkennen kann. In unserem Beispiel wollen wir das nicht machen, sondern zwei getrennte Lexica verwenden, die jeweils das Trainings- und das kombinierte Development- und Test-Set vollständig abdecken.

Ein weiterer Punkt sind mehrfache Einträge im Lexikon, um die lexikale Variabilität von Wörtern zu modellieren. Wir wollen uns hier auf den einfachsten Fall, ein sog. 'kanonisches' Lexikon beschränken. D.h. pro Wort im Lexikon gibt es genau eine kanonische (Standard-)Aussprache.

Der dritte Punkt ist technischer Art: HTK verlangt ein ganz bestimmtes Format für seine Dictionaries und hat außerdem gewisse Beschränkungen, die beim Erstellen berücksichtigt werden müssen:

- Sonderzeichen müssen maskiert werden (dazu gehören z.B. Akzente in der orthographischen Schreibweise
- Die Modell-Bezeichnungen dürfen keine führenden Zahlen haben (das führt zu Problemen bei den Phonemen /2:/, /6/ und /9/)
- Im Dictionary darf kein phonetisches Zeichen vorkommen, das nicht auch durch ein HMM modelliert werden kann. Deshalb müssen seltene Segmente wie /Z/ oder /o/ auf ähnliche Segmente abgebildet werden.[1]
- Das Format für eine Lexikon-Zeile in unserem Beispiel ist

  `<Orthographie> TAB <Aussprache>`

  Wobei <Aussprache> eine Liste von Modellbezeichnungen ist, die durch Leerzeichen getrennt sind.

Zum Beispiel:

```
heute          h OY t @
'Br"ucke'      b R Y k @
```

In der letzten Zeile steht die Orthographie in Hochkommata, weil sie wegen des Umlauts ein Gänsefüßchen enthält.

### Training Dictionary

Zunächst bilden wir eine Liste aller distinktiven Wörter im Trainings-Set:

```
% cd ~/HTK-KURS/GROUP#
% cat TRAIN.REF.mlf | grep -v '^#' | grep -v '^!' | grep -v '^"' \
  | grep -v '^\.$' | sort -u > TRAIN.list
% wc -l TRAIN.list
% less TRAIN.list
```

Die Pipe im Einzelnen: wir suchen aus dem Referenzwort-MLF alle Zeilen, die nicht mit `'#'` beginnen (`grep -v '^#'`), die nicht mit '!' beginnen (`grep -v '^!'`), die nicht mit `'"'` beginnen (`grep -v '^"'`) und die nicht nur ein einzelnen Punkt enthalten (`grep -v '^\.$'`). Dann sortieren wir die Liste, löschen alle mehrfachen Zeilen (sort -u) und schreiben sie nach `TRAIN.list`.

Überlegen Sie sich, warum die einzelnen Schritte notwendig sind. Welche 'Wörter' fehlen jetzt in der Liste? Warum gibt es Wörter wie `"u"s`?

Jetzt brauchen wir die kanonische Aussprache für diese Wortliste. Dazu gibt es glücklicherweise zum Verbmobil-Korpus eine sog. Wortliste ~/HTK-KURS/vm_ger.lex, welche zu jedem Wort bzw.

[1] Oder man macht einen entsprechenden Eintrag in die Phonem-Liste (s. später), so dass /Z/ als sog. virtuelles Phoneme deklariert wird, welches aber eigentlich auf das HMM /S/ verweist.

Wortfragment eine kanonische Aussprache in SAM-PA bereithält. Wir brauchen jetzt also ein Skript das unsere Wortliste TRAIN.list Wort für Wort durchgeht und die richtige Aussprache aus der Wortliste heraussucht.

Das ist nicht ganz trivial, weshalb ich ein AWK-Skript vorbereitet habe, das das für uns erledigt. (Wer neugierig ist, kann sich das Skript mal anschauen; es ist allerdings nicht dokumentiert.)

```
% cd ~/HTK-KURS/GROUP#
% cat TRAIN.list | gawk -f ../condens.awk > TRAIN.dict
% wc -l TRAIN.dict
% less TRAIN.dict
```

Wie Sie bemerken, haben wir jetzt zwar eine Aussprache für jedes Wort des Lexikons, aber die Phonem-Modell-Namen in der zweiten Spalte sind nicht durch Leerzeichen getrennt. Also müssen wir diese Spalte noch parsen. Dazu brauchen wir eine Liste der verwendeten Phonem-Modelle, damit der Parser /a/ von /aU/ unterscheiden kann. Diese Liste habe ich bereits vorbereitet, indem ich das Trainingskorpus nach allen vorkommenden Phonemen abgesucht habe:

```
% less ~/HTK-KURS/PHONEME
% wc -l ~/HTK-KURS/PHONEME
```

Wir brauchen also genau 44 Phonem-Modelle, einschließlich dem 'nicht-sprachlichen-Bereich' /<nib>/ ('Müllmodell'), artikulatorischem Geräusch /<usb>/ und der Pause /<p:>/.

Jetzt parsen wir das Lexikon und erzeugen das entgültig verwendbare Lexikon TRAIN.lex:

```
% cd ~/HTK-KURS/GROUP#
% cat TRAIN.dict | ../create_lex > TRAIN.lex
% wc -l TRAIN.lex
% less TRAIN.lex
```

Was bedeuten die Warnungen, die das Programm 'create_lex' ausgibt?

```
-----------------------------------------------------------------
SIGNALISIEREN SIE DAS ENDE DER UEBUNG - BESPRECHUNG VON FRAGEN
-----------------------------------------------------------------
```

### Test und Development Dictionary

Die Erzeugung des Test- und Development-Lexikons TESTDEV.lex erfolgt ganz analog:

```
% cd ~/HTK-KURS/GROUP#
% cat TEST.REF.mlf DEV.REF.mlf | grep -v '^#' | grep -v '^!' | \
  grep -v '^"' | grep -v '^\.$' \
  | sort -u > TESTDEV.list

% cat TESTDEV.list | gawk -f ../condens.awk > TESTDEV.dict

% cat TESTDEV.dict | ../create_lex > TESTDEV.lex
```

Hätte man die Erzeugung des Dictionaries auch mit einem einzigen Befehl machen können?

### Ergänzung um Pause und Geräusch-Modelle

Die Lexica brauchen noch sieben zusätzliche 'Wörter':

```
Anfangs-Pause     !ENTER
Ende-Pause        !EXIT
Pause             !SIL
Atmen             !BREATH
Lachen            !LAUGH
Artik. Ger.       !ANOISE
Anderes Ger.      !NOISE
```

Wir editieren also die beiden Lexica und fügen die folgenden Zeilen ganz oben ein. ACHTUNG: zwischen den Spalten müssen TABs stehen, keine Leerzeichen!

```
!ANOISE    <usb>
!BREATH    <br:>
!ENTER     <p:>
!EXIT      <p:>
!LAUGH     <la:>
!NOISE     <nib>
!SIL       <p:>
```

Beachten Sie, daß wir hier fünf 'Phoneme' einführen, von denen drei (<nib>,<br:>,<la:>) nicht in der PHONEME Liste vorkommen. Der Grund hierfür ist, daß diese drei Segmente im Bootstrap-Material (siehe nächster Punkt) nicht segmentiert wurden, und daher erst später beim Nach-Training relevant werden.

### Überprüfung

Die nächste interessante Frage ist, ob es denn für jedes der benötigten Phonem-Modelle auch genügend Daten gibt? Zur Beantwortung dieser Frage müssen wir die Anzahl der einzelnen Modelle im gesamten Trainingsmaterial abschätzen, indem wir die Anzahl im Bootstrap-Teil (TRAIN) bestimmen und mit 3 multiplizieren. Der folgenden HLStats Befehl sucht alle Label aus dem Bootstrap-Set heraus, die seltener als 1000mal auftauchen.

```
% cd ~/HTK-KURS/GROUP#
% HLStats -c 1000 ../PHONEME TRAIN.mlf
```

Selbst das seltenste Label /<usb>/ kommt über 200mal vor. Hochgerechnet auf das gesamte Training-Korpus sind dies ca. 600 Vorkommen, was gerade noch für ein Training ausreicht. Wir müssen also keine weiteren Labels herausnehmen. Möglicherweise warnt uns HLStats davor, daß Label-Files leer sind. Das kann passieren, wenn ein Sprecher in einem Turn gar nichts geäussert hat, sondern nur aus Versehen auf die Taste gedrückt hatte (meistens enthält dann dieser Turn ein technisches Geräusch!). Wir nehmen solche Files sicherheitshalber aus den Skripten heraus.

```
---------------------------------------------------------------
SIGNALISIEREN SIE DAS ENDE DER UEBUNG - BESPRECHUNG VON FRAGEN
---------------------------------------------------------------
```

## 2.3 Language Model

Wir haben für die Verbmobil Task kein vorgegebenes Language Modell (LM). Deshalb müssen wir selber eines erstellen. Normalerweise werden Wortfolge-Statistiken nicht nur aus den Sprach-Korpus der Task geschätzt, weil dieses im allgemeinen nicht genug Text enthält. Man nimmt dann zusätzliche, umfangreiche Text-Korpora, die mit der behandelten Task mehr oder weniger zu tun haben. Im Moment stehen uns solche Text-Korpora nicht zur Verfügung, daher beschränken wir uns auf die Texte des Korpus, das sind etwa 230.000 Wörter. Übliche Text-Korpora zur Abschätzung von Language-Modelle haben einen Umfang von mehreren Millionen.

In HTK kann mit dem Tool HLStats ein einfaches Bi-Gram-LM aus vorhandenen Verschriftungen geschätzt werden.

```
% cd ~/HTK-KURS/GROUP#
% cut -f 1 TESTDEV.lex > TESTDEV.words
% HLStats -b LM -o -T 1 TESTDEV.words TRAIN.REF.mlf
% less LM
```

Der Befehl 'cut' erzeugt eine neue Liste mit nur der ersten Spalte des Lexikons. Der Befehl HLStats berechnet aus den Wort-Referenz-Daten des Training-Sets ein sog. Backoff-Language-Model.

Frage: Warum mussten wir eine neue Wortliste TESTDEV.words erstellen? Warum konnten wir nicht einfach die Wortliste TESTDEV.list verwenden, die wir bei der Erstellung des Lexikons schon einmal berechnet hatten?

Das LM muß nun noch in eine HTK-kompatibele Lattice umgewandelt werden (Zur Erinnerung: HTK verwendet statt eines LM immer eine Lattice, d.h. ein Netzwerk aus Wörtern und statistischen Übergangswahrscheinlichkeiten.)

```
% cd ~/HTK-KURS/GROUP#
% HBuild -n LM -T 1 TESTDEV.words LAT
% less LAT
```

Damit sind alle externen Ressoursen für unser Experiment entsprechend aufbereitet. Der nächste Schritt ist das Design von Modell-Strukturen und das Bootstrappen der HMMs.

```
-----------------------------------------------------------------
SIGNALISIEREN SIE DAS ENDE DER UEBUNG - BESPRECHUNG VON FRAGEN
-----------------------------------------------------------------
```