TEILAUTOMATISIERTER WORKFLOW ZUR AUFBEREITUNG GROSSER AUDIODATENMENGEN FÜR SIGNALBASIERTE ANALYSEN

Christoph Draxler[1], *Felicitas Kleber*[1], *Sven Grawunder*[2,3], *Jürgen Trouvain*[4]
[1]*LMU München,* [2]*MLU Halle/Saale,* [3]*MPI EVA, Leipzig,* [4]*UdS Saarbrücken*
draxler@phonetik.uni-muenchen.de

Kurzfassung: Dieser Beitrag stellt eine teilautomatisierte Aufbereitung umfangreicher Audiodatenmengen für phonetisch-linguistische Korpora vor. Im konkreten Projekt handelt es sich um Nachrichtensendungen, die in großer Zahl täglich produziert werden und sich aufgrund sprechstilistischer Ähnlichkeiten besonders gut für linguistische und phonetische Analysen anbieten. Der Workflow umfasst u. a. eine orthographische Transkription mittels automatischer Spracherkennung, deren Prüfung und Vorsegmentierung im webbasierten Transkriptionseditors Octra, der auch das anschließende Schneiden in Einzeldateien ermöglicht, sowie die automatische Segmentierung auf phonetischer Ebene und Speicherung der Daten in diversen Outputformaten mittels der BAS Webservices. Die hier erstmals größere und dokumentierte Verwendung des Octra-Backends wird u.a. durch Bearbeitungszeiten evaluiert.

1 Einführung

Bei der großen Mehrzahl an Audiodateien in Archiven weltweit handelt es sich um nicht textuell erfasste und nicht annotierte Sprachsignale. Trotz ihres großen Informationspotenzials stehen sie damit für Analysezwecke derzeit nur bedingt zur Verfügung. Aufbau und Aufbereitung eines Korpus aus Sendungen des öffentlich-rechtlichen Rundfunks zu wissenschaftlichen Zwecken ist aus verschiedenen Gründen vielversprechend. Zum einen können sie nach dem Prinzip des *fair use* erfolgen, da die Sendungen aufgezeichnet, transkribiert, und gespeichert werden dürfen. Da die Nachrichtensendungen zudem von vornherein für die öffentliche Ausstrahlung bestimmt waren, muss für die phonetische o. a. Analyse der Stimmen kein eigenes Einverständnis der Sprecher/innen eingeholt werden. Zum anderen ermöglicht die bereits erfolgte Archivierung von Nachrichtensendungen in akustisch sehr guter Audioqualität, die synchrone und diachrone Untersuchung seltener Phänomene auf allen linguistischen Ebenen anhand vergleichbarer Sprechstile [1]. Die Aufbereitung für phonetische Analysen stellt immer noch einen immensen Arbeitsaufwand dar, zumal bei Verzicht auf eine valide Annotation bei spezielleren Korpora (wie hier Radio-Nachrichten) viele seltene Phänomene nicht oder falsch erfasst würden. Eine massive Reduktion der manuellen Arbeiten ist jedoch mehr als sinnvoll. Die freie Verfügbarkeit leistungsfähiger automatischer Spracherkennung erlaubt bereits heute die Generierung qualitativ hochwertiger orthographischer Rohtranskripte. Darauf aufbauend wird hier ein teilautomatisierter Workflow zur Transkripterstellung und -aufbereitung anhand aktueller und historischer Radionachrichten präsentiert, der bislang eher manuell erfolgte. Ausgangspunkt ist eine Auswahl von 49 Radionachrichten aus den Jahren 1956 bis 2017, die direkt vom Saarländischen Rundfunk bzw. aus der ARD-Nachrichtenarche [2] stammen.

2 Workflow

Für die Transkription und Datenaufbereitung für die spätere phonetisch-linguistische Untersuchung wird ein soweit wie möglich automatisierter Workflow entwickelt und evaluiert. Dieser Workflow besteht aus den folgenden Schritten: (1) Vorverarbeitung der Audiodateien, (2) Erstellung eines orthographischen Rohtranskripts mittels automatischer Spracherkennung, (3) manuelle Korrektur der Segmente und des Rohtranskripts, (4) Schneiden der Audiodateien und (5), automatische Segmentierung mittels Webdiensten. Ergebnis ist ein Korpus mit nach thematischen Beiträgen geschnittenen Audiodateien und auf Wort-, Laut- und Silbenebene segmentierten Transkriptionen in mehreren Formaten.

2.1 Vorverarbeitung der Audiodaten

Die Dateinamen des Korpus sind unsystematisch: Neben sehr kurzen gibt es auch sehr lange Namen, die neben Nummerierung, Datum und Senderbezeichnung auch aus Sendungstitel und weiteren Angaben bestehen und teilweise auch problematische Sonderzeichen wie '+' enthalten. Für die vorliegende Untersuchung wurden mit einem Python-Skript Leer- und Sonderzeichen in Dateinamen durch den Unterstrich ersetzt bzw. gelöscht, damit die Dateien in den BAS-Webdiensten verarbeitet werden können.

Ein Teil der Audiodaten lag bereits im .wav-Format vor, die restlichen als .mp3-Dateien. Letztere wurden per Kommandozeilenskript mit dem Medienkonversionstool `ffmpeg` automatisch konvertiert, so dass alle Audiodateien im .wav-Format, mit einer Abtrastrate von 44.100 bzw. 48.000 Hz, mit 16 Bit linearer Quantisierung und in mono vorlagen (ca. 1,22 GB, 03:36:54 Dauer, 02:15 pro Sendung).

2.2 Erstellung eines orthographischen Rohtranskripts

Mit dem folgenden Kommandozeilenbefehl wurde die automatische Spracherkennung whisperx (in der Version 3.1.1) aufgerufen ([3]):

```
whisperx --model large-v3 --lang de --compute_type float32 --output_dir results *.wav
```

Um die Dauer der automatischen Spracherkennung zu messen, wurde whisperx für jede Audiodatei separat aufgerufen. Die Spracherkennung benötigte auf dem BAS-Server knapp 02:45 Stunden für 03:37 Stunden Sprachdaten (Echtzeitfaktor ca. 0,76). Ergebnis war für jede Audiodatei ein orthographisches Rohtranskript mit satzartigen Segmenten. Je nach Format enthalten die Transkriptdateien unterschiedlich detaillierte Informationen (Abb. 1):

`.txt` Transkript ohne Zeitmarken
`.srt` Transkript mit Segmentbeginn und -ende in Zeitangaben, mehrzeilige Einträge
`.json` Transkript mit hierarchischer Segmentierung und wortweisem Erkennungsscore

2.3 Manuelle Korrektur der Segmente und des Rohtranskripts

Die automatische Spracherkennung generiert ein in Segmente unterteiltes orthographisches Transkript mit Interpunktion. Für die weitere phonetische Analyse wird das Transkript nach inhaltlichen Kriterien resegmentiert und segmentweise klassifiziert. Dazu müssen Segmentgrenzen angepasst, Marker eingefügt und Text korrigiert werden. Dafür wird der webbasierte Transkriptionseditor Octra [4] verwendet (Abb. 2).

Neben dem üblichen Setzen, Verschieben und Löschen einzelner Grenzen unterstützt Octra auch das Zusammenziehen mehrerer Segmente zu einem langen Segment. Auf diese Weise können automatisch generierte Segmente ihrem Inhalt entsprechend zusammengefasst werden.

```
46
00:04:35,549 --> 00:04:39,271
Saarländische Rundfunk, Nachrichtenredaktion Monika Seel.
```

```
{
  "start": 275.549,
  "end": 279.271,
  "text": " Saarländische Rundfunk, Nachrichtenredaktion Monika Seel.",
  "words": [
      {"word": "Saarländische", "start": 275.549, "end": 275.989, "score": 0.748},
      {"word": "Rundfunk,", "start": 276.009, "end": 276.509, "score": 0.731},
      {"word": "Nachrichtenredaktion", "start": 276.549, "end": 277.77, "score": 0.878},
      {"word": "Monika", "start": 277.81, "end": 278.311,  "score": 0.923},
      {"word": "Seel.", "start": 278.351, "end": 279.271, "score": 0.737}
    ]
},
```

Abbildung 1 – Abspann der Nachrichtensendung vom 02.10.1990 auf SR1 im automatisch generierten Transkript im .srt-Format mit Segmentgrenzen (oben) und im .json-Format mit Segment- und Wortgrenzen sowie einem Erkennungs-Score (unten).

Die Klassifikation der Segmente nach Textsorte basiert auf sog. *Sendeelementen* (Tab. 1, [5]). Diese Sendeelemente dienen nicht nur der inhaltlichen Gliederung einer Sendung, sondern stehen auch für bestimmte Sprechweisen.

Tabelle 1 – Sendeelemente (Textsorten).

Code	Beschreibung
TIM	Zeitansage
LOC	Ortsangabe (’Berlin.’) oder Thema (’Bundeshaushalt’, ’Sport.’)
COR	Meldung, klass. Nachricht, Core-News
REP	Bericht, Report (oft vorproduzierter Einspieler)
INT	Anmoderation/Intro (zu einem nachfolgenden Bericht/Einspieler)
OUT	Abmoderation/Outro (’Das war Anne Seiler aus Bonn’ nach dem Bericht etc.)
OTO	O-Ton (Bericht oder Interviewausschnitt mit Atmo-Hintergrund)
SPO	Sport
WET	Wetterbericht/-vohersage/-interview
TRA	Verkehr
STO	Börsennachrichten (oft mit Atmo)
TOP	Übersicht (DLF: ’Die Themen....’)
PAC	Jingle/Senderkennung, akustische Trenner, Pausen- bzw. Zeitzeichen, akustische Verpackung

2.4 Transkriptionskonventionen

Die Transkriptionskonventionen unterstützen eine schnelle und konsistente Transkription:

- Wortwörtlich verschriften, Standardorthographie verwenden, Groß- und Kleinschreibung abhängig von der Wortart, nicht der Position im Text.
- Interpunktion wie von der Spracherkennung vorgegeben.
- Abkürzungen als Wort in Großbuchstaben, wenn sie auch so gesprochen werden (z. B. EON), ansonsten in einzelnen Großbuchstaben (z. B. R W E).
- Ziffern und Zahlen wortwörtlich ausschreiben (z. B.drei Komma drei), Zahlen über 100 trennen (z. B. neunzehn hundert zweiundsiebzig).
- Codes für Sendeelemente stehen in spitzen Klammern (z. B. $< COR >$); pro Segment gibt es genau ein Sendeelement, das stets an erster Position steht.

Abbildung 2 – Automatisch generiertes Rohtranskript in der 2D-Ansicht in Octra. Grün hinterlegte Segmente enthalten Transkriptionstext, rote Segmente sind leere Lückensegmente. Die Segmentgrenzen wurden von whisperx gesetzt. Diese Grenzen folgen z. T. so kurz aufeinander, dass sie im Editor nur beim Heranzoomen getrennt dargestellt werden (vgl. Abb. 5).

Octra unterstützt eine projektspezifische formale Überprüfung von Transkripten. Auf diese Weise wird sichergestellt, dass nur formal korrekte Transkripte gespeichert werden.

Zur Verwaltung des Workflows wurde im Octra-Backend [6] ein eigenes Projekt eingerichtet und wurden sog. *Konfigurationen* angelegt. Eine Konfiguration definiert den Typ eines sog. *Task*. Sie spezifiziert z. B. welche Werkzeuge verfügbar und welche Tastaturkürzel aktiviert sind und wie formal korrekte Transkripte aussehen.

Ein konkreter Task besteht aus einem Dateipaar, bestehend aus einer Audio- und der dazugehörigen Segmentdatei. Das Aufgabenfenster im Octra-Backend zeigt den aktuellen Bearbeitungsstand eines Projekts. Tasks sind in einer Liste angeordnet, der jeweilige Status (z. B. *frei*, *in Arbeit*, *erledigt*) ist sofort sichtbar (Abb. 3).

Im aktuellen Projekt wurde ein zweiter Korrekturvorgang definiert, weil sich die Transkriptionsrichtlinien seit dem ersten Durchgang geändert hatten. Im Octra-Backend wurde diese zweite Korrektur als eine mit der ersten Korrektur verknüpfte Aufgabe definiert, bei der die Ausgabe der ersten Korrektur die Eingabe für die zweite war. In der Liste der Aufgaben ist eine solche Verknüpfung von Aufgaben durch mehrere Aufgaben in einer Zeile erkennbar (Abb. 3).

Nach Abschluss der manuellen Korrektur wurde das Projekt exportiert. Das Octra-Backend erlaubt eine flexible Konfiguration der zu exportierenden Dateien. In einem Dialogfenster wird angegeben, welche Audio- und Transkriptdateien, Protokolldateien der einzelnen Transkriptionsdurchläufe sowie Daten der Bearbeiter exportiert werden sollen.

Der Exportvorgang legt auf dem Rechner ein mit dem Projektnamen und dem aktuellen Datum benanntes Projektverzeichnis an. Dieses enthält in je eigenen Unterverzeichnissen die Ausgangs- und Ergebnisdateien sowie Dokumentations- und sonstige Dateien.

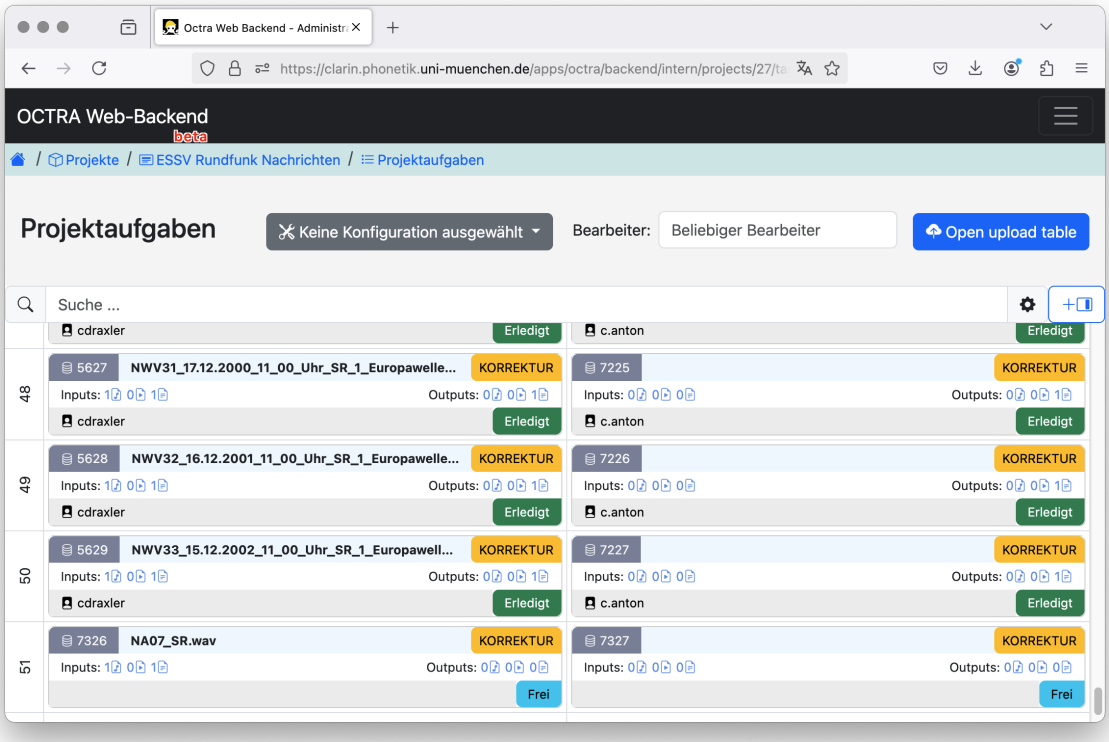

Abbildung 3 – Übersicht der verfügbaren Aufgaben mit Status-Information. Eine Zeile steht für eine Aufgabe, die verknüpften Arbeitsschritte einer Aufgabe stehen in einer Zeile, eine Spalte repräsentiert einen Arbeitsschritt.

2.5 Schneiden der Audiodateien

Octra bietet ein Werkzeug zum Schneiden von Signalen. Damit wird ein Signal entsprechend den Segmenten geschnitten und es wird eine Schnittliste angelegt (Tab. 2). Diese ist besonders dann nützlich, wenn statt der heruntergesampelten Quelldatei, wie sie Octra verwendet, Dateien mit einer höheren Samplerate geschnitten werden sollen. Zum Schneiden der Audiodateien wurde ein eigenes Projekt angelegt, in das Dateipaare bestehend aus den Original-Audiodateien sowie den manuell korrigierten Transkriptdateien importiert wurden.

Tabelle 2 – Schnittliste einer Audiodatei (zur besseren Lesbarkeit ediert).

Fragment	Quelle	Start (s)	Dauer (s)	Transkript
NA03_SR_1_0001.wav	NA03_SR_1.wav	0	6	PAC SR eins …
NA03_SR_1_0002.wav	NA03_SR_1.wav	6	8.898	TOP Anschlussfinanzierung …
…	…	…	…	…
NA03_SR_1_0021.wav	NA03_SR_1.wav	235.865	5.108	OUT soweit die …

2.6 Automatische Segmentierung

Dieser anschließende Arbeitsschritt erfolgt außerhalb des Octra-Backends und verwendet die geschnittenen Audiodateien. Für die weitere Analyse sollten diese geschnittenen Signaldateien automatisch auf Wort-, Silben- und Lautebene segmentiert werden. Mit einem Pipeline-Webdienst des BAS kann diese Segmentierung generiert werden [7]: die Pipeline 'G2P → MAUS → PHO2SYL' benötigt ein Dateipaar aus Audio- und Textdatei als Eingabe und erstellt

```
def callPipeline (audiopath, textpath, language, outformat, pipeline):
  (_, audiofile) = os.path.split(audiopath)
  (_, textfile) = os.path.split(textpath)

  url = "https://clarin.phonetik.uni-muenchen.de/BASWebServices/services/runPipeline"
  formdata = {
    "SIGNAL": (audiofile, open(audiopath, "rb"), "audio/x-wav"),
    "TEXT": (textfile, open(textpath, "r"), "text/txt"),
    "LANGUAGE": (None, language),
    "OUTFORMAT": (None, outformat),
    "PRESEG" : (None, "true"),
    "PIPE" : (None, pipeline)
  }

  result = requests.post(url, files=formdata)
  if result.ok:
    tree = ET.fromstring(result.text)
    downloadLink = tree.find('downloadLink').text
    if downloadLink:
      downloadResult = requests.get(downloadLink)
      if downloadResult.ok:
        return downloadResult.text

  return None
```

Abbildung 4 – Python-Code für den Aufruf des Pipeline-Dienstes G2P→MAUS→PHO2SYL. Ergebnis ist der Text der Segmentation im gewünschten Ausgabeformat.

eine Segmentierung in einem vorgegebenen Ausgabeformat.

BAS-Webdienste lassen sich über eine grafische Schnittstelle oder per API-Aufruf aus einem Skript heraus aufrufen. Letzteres ermöglicht eine automatisierte Ausführung, die zwar wegen des zusätzlichen Kommunikationsaufwands langsamer ist – dieser Nachteil wiegt aber den der automatischen Ausführung in der Regel auf.

Im aktuellen Projekt generiert ein Python-Skript zunächst aus der Schnitttabelle für jede Audiodatei die passende Transkriptdatei, ruft den Webdienst mit diesem Dateipaar auf und speichert das Ergebnis auf dem lokalen Rechner. Abb. 4 zeigt den Python-Code für den API-Aufruf des Webdiensts. Die Pipeline liefert als Ergebnis Dateien im BAS-Partitur-Format. Diese wurden anschließend – ebenfalls per Python-Skript – mit dem Webdienst AnnotConv verlustfrei ins Praat-Format TextGrid ([8]) bzw. annot.json (für das Emu-SDMS [9]) konvertiert[1].

3 Ergebnisse

Nach Beendigung der Transkription und dem Export liegen die Audio- und Transkriptdateien in einem Verzeichnis auf dem lokalen Rechner. Dazu kommt pro Nachrichtensendung ein eigenes Verzeichnis mit den geschnittenen Signalen sowie der Schnittliste. Das Projektverzeichnis dient zum einen als Datensicherung, zum anderen sind die darin enthaltenen Transkripte in `.txt`-Format die Grundlage für die Berechnung der Wortfehlerrate der automatischen Transkription.

3.1 Wortfehlerrate

Die Formate der von der automatischen Spracherkennung generierten und die der manuellen Transkripte sind verschieden. Eine aussagefähige Berechnung der Wortfehlerrate erfordert daher eine Normalisierung der Texte: Entfernen der Sendeelemente-Codes, Löschen von Satz- und Sonderzeichen, Kleinschreibung.

Für die Berechnung der Wortfehlerrate wurden die automatisch generierten Transkripte so-

[1] Für den Import der annot.json-Dateien in eine vollständige Emu-Datenbank muss einmal eine Datenbank-Konfigurationsdatei erstellt werden. Dies ist durch den Aufruf der Pipeline-Verarbeitung über die grafische Schnittstelle und das anschließende Sichern als zip-Archivdatei möglich.

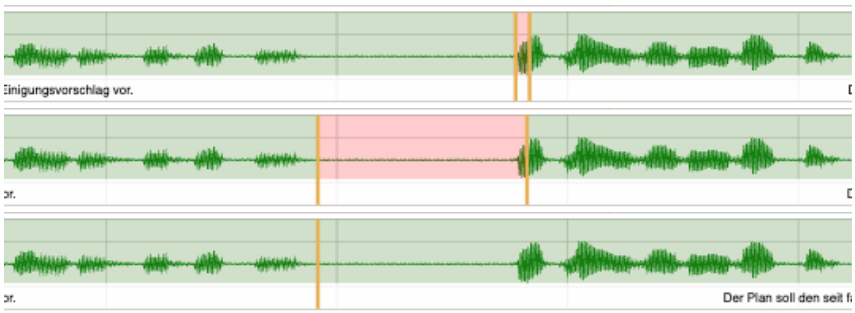

Abbildung 5 – Problematische Segmentgrenzen der Spracherkennung: nah beieinander liegende Grenzen im Wort (oben). Korrektur durch Verschieben der linken Grenze an den Anfang der Pause (Mitte) und Löschen der rechten Grenze (unten).

wie die beim Export der manuell korrigierten Transkripte erzeugten Textdateien verwendet. Die Texte wurden normalisiert und mit dem Paket `jiwer` ([10]) in einem Python-Skript ausgewertet. Die Wortfehlerrate liegt bei 8.85%. Dieser Wert scheint zunächst hoch, lässt sich aber damit erklären, dass Nachrichten recht viele Zahlen enthalten, die in automatisch generierten Transkripten als numerische Angaben und in manuell erstellten Transkripten in ausgeschriebener Form stehen. Orts- und Personennamen wurden durchwegs gut erkannt (auch solche von historischer oder politischer Bedeutung, z. B. 'Douaumont', 'San Son'), Sendernamen oder Bezeichnungen mit besonderer Schreibweise hingegen schlechter (z. B. 'Studiowelle Saar' als 'Studio Velesa' oder 'Studio Velesar', 'Saar-Lor-Lux' als 'Saarlouis-Lux').

3.2 Segmentgrenzen

Manuell korrigiert werden mussten insbesondere von der Spracherkennung generierte Satzgrenzen, die häufig in sehr kurzem Abstand nacheinander und oft nicht in Sprechpausen, sondern im Wort gesetzt wurden (siehe Abb. 5).

3.3 Klassifikation der Segmente

Eine konsistente Klassifikation der Segmente ist kurzfristig durch Schulung und den Vergleich mit Mustertranskriptionen zu lösen, mittelfristig sicherlich zuverlässig mittels KI.

Im Schnitt benötigten die zwei Transkribenden (der Erstautor sowie eine studentische Hilfskraft) 1.720 bzw. 1.190 Sekunden für eine Nachricht mit einer durchschnittlichen Dauer von 265 Sekunden (Echtzeitfaktor ca. 6,9 bzw. 4,3). Diese Angaben sind vorläufig, da die Transkriptionen für die Transkribenden die ersten dieser Art waren und die Transkripte aufgrund von Konventionsanpassungen während des Projektes teils überarbeitet werden mussten.

4 Zusammenfassung und Ausblick

Das vorliegende Projekt ist die erste größere und dokumentierte Verwendung des Octra-Backends. Während der Arbeit am Korpus wurde eine neue Version des Backends freigegeben, in der nun auch die automatische Spracherkennung automatisiert als Hintergrundprozess ausgeführt und somit in den Workflow eingebunden werden kann.

Die manuelle Korrektur kann zwar nicht automatisiert, aber immerhin maschinell unterstützt werden: so scheint es möglich, die Klassifikation von Segmenten nach Sendeelementen automatisch durchzuführen – für ein entsprechendes Training liegen jedoch aktuell noch nicht ausreichend viele Daten vor.

Als weitere Verbesserungen sind zusätzliche Export-Optionen geplant: (1) Erstellen einer vollständigen Emu-Datenbank und (2) automatisches Schneiden der Audiodateien mit gleich-

zeitigem Generieren von Transkriptdateien aus den Segmenten für die Weiterverarbeitung in z. B. BAS Webdiensten. Ein grundsätzliches Problem ist, dass Octra aktuell Audiodaten intern automatisch auf 16.000 bzw. 22.050 Hz heruntersampelt, um auch sehr lange Audiodateien verarbeiten zu können. Beim Schneiden werden die Signalfragmente aus dem internen Audiosignal extrahiert und haben somit die gleiche Abtastrate. Eine Import-Option, Audiodaten intern auch in der Originalqualität zu verwenden, ist daher ebenfalls in Vorbereitung.

Danksagung

Ein besonderer Dank geht an die studentische Hilfskraft Camilla Anton, die in sehr kurzer Zeit die manuelle Korrektur der Transkriptionen durchgeführt hat. Die Arbeiten am Octra Backend werden teilweise aus dem NFDI Vorhaben Text+ (DFG-Fördernummer 460033370) finanziert.

Literatur

[1] KUPIETZ, M., C. BELICA, H. KEIBEL, und A. WITT: *The German reference corpus DeReKo: A primordial sample for linguistic research*. In N. CALZOLARI, K. CHOUKRI, B. MAEGAARD, J. MARIANI, J. ODIJK, S. PIPERIDIS, M. ROSNER, und D. TAPIAS (Hrsg.), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), Valletta, Malta, 2010. URL `https://aclanthology.org/L10-1285/`.

[2] SCHWIESAU, D., S. GRAWUNDER, und I. BOSE: *Die Nachrichtenarche der ARD*. In I. BOSE und D. SCHWIESAU (Hrsg.), *Nachrichten Schreiben, Sprechen, Hören: Forschungen Zur Hörverständlichkeit von Radionachrichten*, S. 147–155. Frank & Timme, Berlin, 2011.

[3] BAIN, M., J. HUH, T. HAN, und A. ZISSERMAN: *WhisperX: Time-Accurate Speech Transcription of Long-Form Audio*. In *Proc. Interspeech 2023*. Incheon, Korea, 2023.

[4] PÖMP, J. und C. DRAXLER: *Octra – A configurable browser-based editor for orthographic transcription*. In *Proceedings Phonetik und Phonologie*, S. 145–148. Berlin, 2017.

[5] GRAWUNDER, S.: *Die Erforschung des Sprechens mittels Nachrichtenkorpora: Die Nachrichtenarche der ARD*. In I. BOSE und D. SCHWIESAU (Hrsg.), *Nachrichten Schreiben, Sprechen, Hören: Forschungen zur Hörverständlichkeit und Radionachrichten*, S. 147–155. Frank & Timme, Berlin, 2011.

[6] DRAXLER, C. und J. PÖMP: *Octra Backend - eine skalierbare Infrastruktur für Transkriptionsprojekte*. In *Proc. ESSV 2024*. TUDpress, Regensburg, 2024.

[7] KISLER, T., F. SCHIEL, und H. SLOETJES: *Signal Processing Via Web Services: The Use Case WebMAUS*. In *Proceedings Digital Humanities*, S. 30–34. Hamburg, 2012.

[8] BOERSMA, P.: *Praat, a System for doing Phonetics by Computer*. *Glot International*, 5(9/10), S. 341–345, 2001.

[9] WINKELMANN, R., J. HARRINGTON, und K. JÄNSCH: *Emu-SDMS: Advanced speech database management and analysis in R*. *Computer Speech & Language*, 45, S. 392–410, 2017.

[10] VAESSEN, N.: *JiWER: Similarity measures for automatic speech recognition evaluation*. *Python Package Index*, 2018.