EINE WEBBASIERTE EXPERIMENTIERUMGEBUNG MIT SPRACHDATENBANK UND SPRACHERKENNUNG

Stephan Euler

Technische Hochschule Mittelhessen, Fachbereich MND
Wilhelm-Leuschner-Straße 13, 61169 Friedberg
Stephan.Euler@mnd.thm.de

Abstract:
Im aktuellen Standard HTML5 wurde die Unterstützung von Audio-Aufnahme und Wiedergabe aufgenommen. Davon ausgehend wird in diesem Beitrag ein Konzept vorgestellt, um sowohl das Sammeln von Sprachdaten als auch die Durchführung von Erkennungsexperimenten in eine Web-Anwendung zu integrieren. Die entsprechenden grundlegenden Funktionalitäten sind in einem ersten Prototyp realisiert. Über erste Erfahrungen aus dem Einsatz in der Lehre wird berichtet. Anschließend werden vielfältige Möglichkeiten zum weiteren Ausbau vorgestellt.

1 Einleitung

Das Trainieren und Testen von Systemen zur automatischen Spracherkennung erfordert entsprechende Sprachdaten. Dies beinhaltet zunächst die Audiosignale mit den zugehörigen Transkriptionen. Für die Verarbeitung müssen die Audiodateien häufig noch in mehreren Varianten beispielsweise mit unterschiedlichen Abtastraten vorgehalten werden. Hinzu kommen Dateien mit den berechneten Merkmalsvektoren, Modellparametern, Grammatiken, Verlaufsinformationen sowie schließlich den Erkennungsergebnissen. In diesem Beitrag wird ein Ansatz beschrieben, bei dem alle Daten zentral auf einem Webserver gehalten werden. Ziel ist dabei der Einsatz der Anwendung in der Lehre. Die übersichtliche Struktur soll es beispielsweise ermöglichen, in einer Veranstaltung zu dem Thema „Automatische Spracherkennung“ den Zyklus von Datensammlung, Modelltraining und Erkennertest durchzugehen.

2 Anforderungen

Aus dem vorgesehenen Einsatz als Experimentierumgebung ergeben sich eine Anzahl von notwendigen sowie wünschenswerten Anforderungen. Zunächst soll es möglich sein, Aufnahmen direkt im Browser aufzusprechen. Bekannte Lösungen beruhen auf zusätzlichen Technologien wie etwa Flash [1]. Mit dem neuen HTML5-Standard wurde allerdings das Element <audio> und eine entsprechende Schnittstelle (Web Audio API [2]) für Audio-Verarbeitung eingeführt. Die Anwendung soll diese Schnittstelle verwenden und ohne zusätzliche Technologien auskommen. Gleichzeitig soll es auch möglich sein, vorhandene Audio-Dateien in das System zu laden. Dabei sollen die gängigen Audio-Formate unterstützt werden.

Zu den Aufnahmen gehören Transkriptionen. Zumindest die Transkription auf Wortebene ist bereits für die einfachsten Erkennungsexperimente notwendig. Zusatzinformationen z. B. über Sprecher, Mikrophon oder Aufnahmebedingungen sind im Prinzip wünschenswert. Bei den personenbezogenen Daten ist dabei der Datenschutz zu berücksichtigen.

Im ersten Schritt soll ein Erkenner für ganze Wörter auf der Basis von Hidden Markov Modellen (HMM) integriert werden. Über die Webseite sollen Training und Test angestoßen werden können. Die Erkennungsergebnisse sollen zur Dokumentation ebenfalls in der Datenbank abgelegt werden.

Die Anwendung ist auch als Basis für Erweiterungen durch Projekte von Studierenden vorgesehen. Daher sollen möglichst nur frei verfügbare Technologien und offene Standards verwendet werden.

3 Realisierung

3.1 Architektur

Kernstück des Systems[1] ist eine relationale Datenbank, in der die Informationen zu den Dateien abgelegt sind. Die Web-Anwendung basiert auf Laravel, einem der aktuellen Web-Frameworks für die Programmiersprache PHP [3]. Laravel folgt dem *model view controller* Muster (MVC) mit klarer Trennung zwischen Datenmodell, Präsentation und Programmsteuerung. Für ein flexibles Layout, das sich automatisch den unterschiedlichen Möglichkeiten der verschiedenen Endgeräte anpasst, wird ergänzend das CSS-Framework Bootstrap eingesetzt.

Die Live-Aufnahme wird in JavaScript gemäß der Web Audio API realisiert. Damit kann man direkt im Browser (zumindest in denen, die die entsprechenden HTML5 Möglichkeiten unterstützen) eine Aufnahme erstellen. Allerdings ist die Unterstützung dieser API in den Browsern noch nicht vollständig. So ist bisher nicht gelungen, eine andere Abtastrate als die vorgegebenen 44 kHz einzustellen.

Die Integration der Spracherkennung in die Web-Anwendung erfordert eine Anbindung an die Datenbank. Der Programme zum Training und zur Erkennung sollen Informationen wie Liste der Aufnahmen und Transkriptionen aus der Datenbank lesen und auch Ergebnisse zurück schreiben. Daher werden eigene Anwendungen verwendet, bei denen diese Integration einfacher ist als bei externe Programme wie etwa HTK. Die Anwendungen sind in Java programmiert, so dass sie weitgehend Plattform-unabhängig sind und sich problemlos auf dem Server ausführen lassen.

Besonderer Wert wird auf die Offenheit des Systems gelegt. Daher wird in allen Dateien – mit Ausnahme der Audiodateien – durchgängig die Auszeichnungssprache XML (eXtensible Markup Language) verwendet. Damit sind alle Dateien direkt lesbar. Gleichzeitig lassen sich XML-Dokumente bei Bedarf leicht in andere Formate transformieren. In den meisten Fällen werden die Daten zurzeit beim Öffnen im Browser direkt als XML-Dokumente dargestellt. Es ist geplant, dies durch geeignete z. B. grafische Darstellungen zu ergänzen.

3.2 Datenmodell

Im System sind drei Rollen für die Benutzer vorgesehen. Zunächst ist jeder Besucher und jede Besucherin der Anwendung ein *Gast*. Gäste können alle Dateien ansehen beziehungsweise anhören. Sie dürfen aber keinerlei Änderungen vornehmen. Durch eine Registrierung kann ein Gast zu einem *Nutzer* werden. Zur Registrierung muss eine Email-Adresse angegeben werden, an die ein Bestätigungslink geschickt wird. Nutzer können Aufnahmen im System speichern und Erkennungstests durchführen. Alle hinzugefügten Daten sind mit dem jeweiligen Nutzer verbunden und können von ihm selbst wieder gelöscht werden. Schließlich ist im Datenmodell noch die Rolle *Administrator* mit weitergehenden Rechten vorgesehen. Allerdings wurden bisher noch keine entsprechenden Möglichkeiten implementiert.

Ein Benutzer kann mehrere so genannte Sets für die Unterteilung seiner Aufnahmen anlegen. Beim Anlegen eines Sets entscheidet man, ob die darin enthaltenen Aufnahmen als Referenz-

[1] aktuelle Version http://wan15.mnd.thm.de/laravel/public/

oder als Test-Daten verwendet werden. Jede Aufnahme hat eine Transkription, die wiederum aus einem oder mehreren Wörtern besteht.

3.3 Ablauf Aufnahme einstellen

Weitere Aufnahmen lassen sich nach folgendem Ablauf in das System einzufügen: zunächst wählt der Benutzer einen eigenen Set aus. Dann besteht die Auswahl zwischen den beiden Möglichkeiten Hochladen einer Datei und Live-Aufnahme. In beiden Fällen muss noch die zugehörige Transkription eingetragen werden. Dann schließt der Benutzer mit dem Hochladen den Vorgang ab.

Auf dem Server wird zunächst ein eindeutiger Dateiname gebildet. Dazu wird bei Bedarf ein zufälliges Kürzel an den vorhandenen Namen angehängt. Unter diesem Namen wird die Datei in das zum Set gehörende Verzeichnis gespeichert. Für die Experimente zur Spracherkennung wird mit dem Programm Tool SoX[2] (Sound eXchange) eine Version mit den Standardeinstellungen Mono, 16 kHz Abtastrate und WAVE-Format erstellt. Weiterhin wird eine Datei mit daraus berechneten Merkmalsvektoren (derzeit Mel Frequency Cepstral Koeffizienten) abgelegt. Die drei Dateien – Original, normalisierte Audiodatei und Merkmalsdatei – liegen parallel in drei Verzeichnis-Ästen. Bei einer künftigen Erweiterung auf z. B. eine andere Abtastrate oder andere Merkmale müssten dann nur weitere Äste eingefügt werden. Schließlich wird noch geprüft, ob die angegebene Transkription bereits in der Datenbank vorhanden ist. Falls nicht, wird sie eingetragen und – sofern notwendig – wird auch die Wortliste erweitert.

3.4 Ablauf Erkennungsversuche

Derzeit ist ein Erkenner für ganze Wörter auf der Basis von Hidden Markov Modellen (HMM) mit einem gemeinsamen Satz von Dichtefunktionen im System integriert. Über die entsprechende Webseite kann sowohl das Training als auch der Erkennertest gestartet werden. Das Training selbst besteht aus zwei Phasen. In der Initialisierung werden auf der Basis einer linearen Segmentierung aller Referenzäußerungen Mittelwerte und Varianzen der Dichtefunktionen ermittelt und die Markov Modelle werden mit Anfangswerten belegt. Im eigentlichen Training werden dann die Gewichte der Dichtefunktionen mittels Viterbi-Training optimiert.

Abbildung 1 zeigt den entsprechenden Ausschnitt aus der Webseite zum Starten eines Trainingsdurchlaufs. Auf dem Server werden die entsprechenden Java-Anwendungen gestartet. Sie erhalten als Argumente Informationen über den aktuellen Benutzer und gegebenenfalls das Start-Modell. Die Anwendungen holen sich die Liste der Äußerungen und die zugehörigen Transkriptionen aus der Datenbank und tragen am Ende wiederum das neue Modell in die entsprechende Tabelle ein.

Da ein Trainingsdurchgang durchaus einige Zeit dauern kann, werden bereits im Verlauf Informationen für den Fortschritt im Browser angezeigt. Dazu schreibt die Trainingsanwendung regelmäßig (zur Zeit nach jeweils 50 verarbeiteten Äußerungen) die Nummer des Durchgangs, die Gesamtzahl der verarbeiteten Äußerungen und die Uhrzeit in eine weitere Datenbank-Tabelle. Auf Client-Seite wird mittels AJAX (Asynchronous JavaScript and XML) dieser Stand periodisch abgefragt und damit das Status-Element der Webseite aktualisiert. Die Ausgaben der Trainer-Anwendung werden während der Ausführung auf dem Server gesammelt und nach Ende ebenfalls an den Browser geschickt.

[2] sox.sourceforge.net

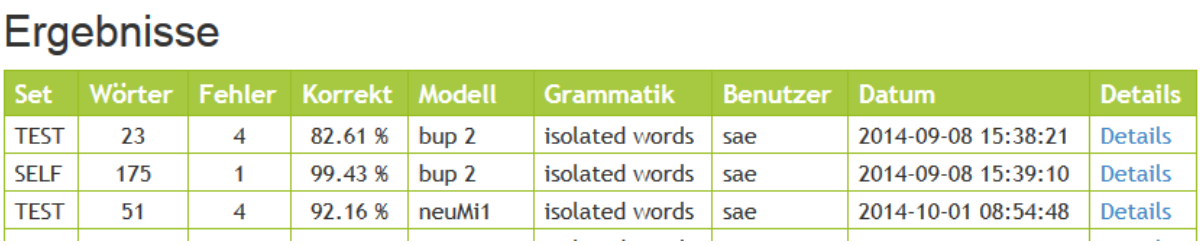
Abbildung 1 Auswahl für Training

Ähnlich ist der Ablauf beim Erkennertest. Man wählt ein Modell aus und startet dann die Erkennung von Test- oder Referenz-Daten (Abbildung 3). Dabei ist schon die Auswahl einer Grammatik vorgesehen. Allerdings ist bisher nur die Erkennung von einzeln gesprochenen Wörtern implementiert. Während die Erkennung auf dem Server abläuft, wird im Browser der Fortschritt angezeigt. Die Verlaufsinformationen und Erkennungsergebnisse werden wie beim Training nach Abschluss gesammelt angezeigt. Abbildung 3 zeigt ein entsprechendes Beispiel.

Abbildung 2 Auswahl für Erkennertest

Abbildung 3 Darstellung der Ergebnisse im Browser

Eine Zusammenfassung der Erkennungsergebnisse wird wiederum in der Datenbank abgelegt. Abbildung 4 zeigt den Anfang der entsprechenden Darstellung. Über den Verweis *Details* können genauere Informationen abgerufen werden. Diese werden für jeden Erkennungslauf in einer eigenen XML-Datei abgelegt. Beispielsweise sieht man in der Detaildarstellung alle falsch erkannten Äußerungen.



Ergebnisse

Set	Wörter	Fehler	Korrekt	Modell	Grammatik	Benutzer	Datum	Details
TEST	23	4	82.61 %	bup 2	isolated words	sae	2014-09-08 15:38:21	Details
SELF	175	1	99.43 %	bup 2	isolated words	sae	2014-09-08 15:39:10	Details
TEST	51	4	92.16 %	neuMi1	isolated words	sae	2014-10-01 08:54:48	Details

Abbildung 4 Anzeige der Erkennungsergebnisse

Die Modellparameter werden in einer XML-Datei abgespeichert. Die Datenbank enthält eine Liste aller Modelle. Für Tests unabhängig von Web-Anwendung können die Modell-Dateien heruntergeladen werden. Mit der Java-Anwendung FbRecognizer steht ein einfacher Erkenner zur Verfügung. Abbildung 5 zeigt beispielhaft die Erkennung des Wortes *Montag*.

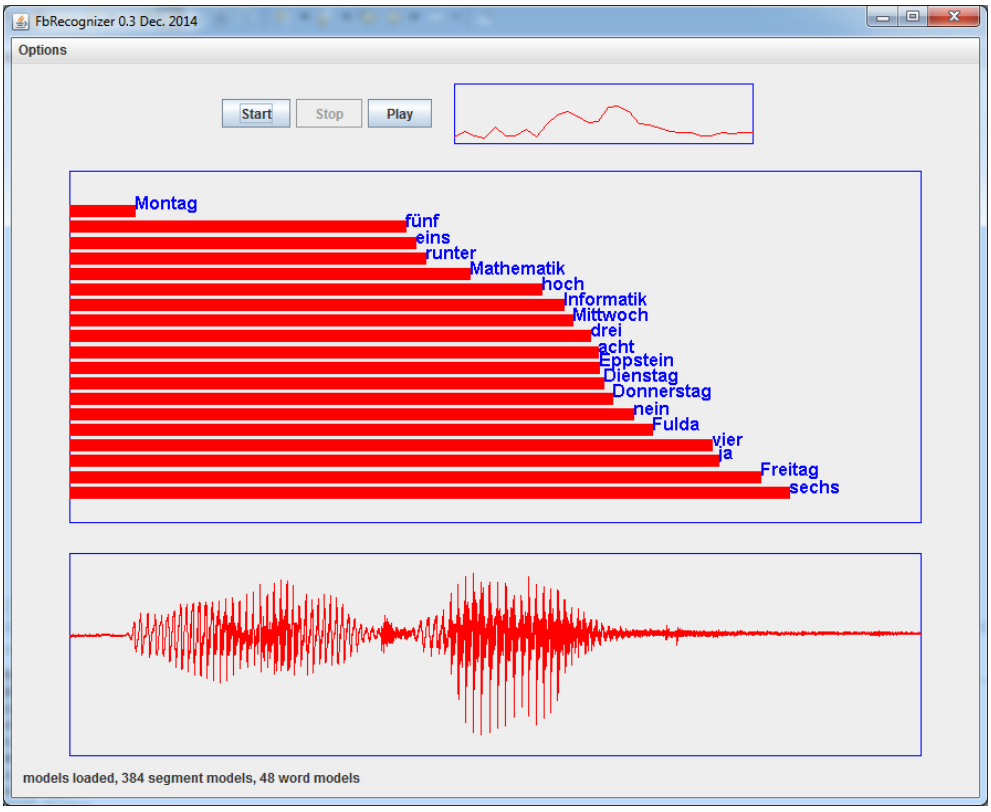


Abbildung 5 Beispielerkennung in FbRecognizer

4 Erster Einsatz in einer Vorlesung

Im Wintersemester 2014/15 wurde die Anwendung erstmals in der Vorlesung Aud*iovisuelle Medien* im Studiengang Medieninformatik an der TH Mittelhessen eingesetzt. Diese Veranstaltung beinhaltet unter anderem eine kurze Einführung in das Thema

Spracherkennung. Anhand eines praktischen Beispiels sollte die Vorgehensweise zum Trainieren eines Erkenners veranschaulicht werden.

Als Vokabular wurden die sieben Wochentage gewählt. Die Studierenden testeten mit der Anwendung FbRecognizer die Erkennung dieser Wörter. Zu diesem Zeitpunkt enthielt die Datenbank nahezu ausschließlich Aufnahmen des Autors, so dass die Erkennungsquote in den meisten Fällen nicht sehr gut war. FbRecognizer speichert alle Spracheingaben ab. Diese Dateien wurden von den Teilnehmern gesammelt und in die Datenbank eingefügt. Mit diesen erweiterten Daten wurden die Erkennungsmodelle in mehreren Stufen neu trainiert. Die Erkennungsquote zeigte das erwartete Verhalten: durch die Hinzunahme von neuen Sprechern und Sprecherinnen sank sie anfangs ab und stabilisierte sich erst langsam mit zunehmendem Umfang der Trainingsdaten.

5 Weitere Arbeiten

Die Anwendung hat den Stand eines ersten Prototyps erreicht. Die grundlegenden Abläufe funktionieren und ein erster, kleiner Bestand an Aufnahmen liegt vor. Im Weiteren sind zunächst kleinere Verbesserungen und Erweiterungen der Kernfunktionalitäten geplant. Darüber hinaus bieten sich mittel- und langfristig eine ganze Reihe von weitergehenden Erweiterungen an.

Die Sprachaufnahme im Browser ist zwar bereits möglich, allerdings fehlt noch die automatische Erkennung von Anfang und Ende der Sprache innerhalb der Aufnahme. Zeitweise war daher ein Audio-Editor in die Anwendung integriert, mit dem man vor dem Hochladen Pausen abscheiden konnte. Allerdings wurde dieser Editor aufgrund von Kompatibilitätsproblemen vorerst wieder deaktiviert. Fast alle Aufnahme wurden bisher nicht direkt eingesprochen sondern als bestehende Dateien einzeln hochgeladen. Hier sind Erleichterungen zum gleichzeitigen Importieren mehrerer Dateien sinnvoll.

Mittelfristig sollen auch ergänzende Informationen über Aufnahmebedingungen und Sprecher beziehungsweise Sprecherin gespeichert werden. Für die Speicherung dieser eher unstrukturierten Daten erscheint eine relationale Datenbank weniger geeignet. Daher ist der Einsatz einer zusätzlichen dokumentenbasierten Datenbank geplant. Die Daten werden dabei als Schlüssel-Wert-Paare abgelegt. Damit wäre es auch möglich, weitere Informationen wie z. B. Transkriptionen auf unterschiedlichen Ebenen abzulegen. Verbunden mit der Speicherung und Pflege solche Metadaten ist eine erweiterte Suchfunktion.

Beim eingebauten Erkenner ist noch viel zu tun:

- Optimierung der Merkmalsextraktion (Einbau Delta-Cepstrum) und Modellstrukturen (z. B. Anzahl der Zustände der Markov-Ketten)
- Berücksichtigung von Grammatiken
- Umstellung auf Phonem-basierte Erkennung

Hierbei ist die Frage, inwieweit es sinnvoll ist den eigenen Erkenner weiter zu entwickeln oder zu einem der frei verfügbaren Systeme zu wechseln. Als Thema für ein studentisches Projekt bietet sich die grafische Darstellung der Modelle im Browser an. Diese könnte noch zu einem Editor erweitert werden, mit dem man beispielsweise Zuständen splitten oder kombinieren könnte.

Neben den technischen Aspekten bleibt die Aufgabe, mehr Sprachdaten in das System zu bringen. Aktuell (Stand Ende Januar 2015) sind es nur 61 Wörter und etwa 700 Aufnahmen. Dabei ist zu beobachten, wie das System mit wachsender Größe skaliert und ob beispielsweise der Umfang von Ausgaben oder die Listen-Darstellungen angepasst werden müssen.

6 Zusammenfassung

In diesem Beitrag wurde das Konzept für eine web-basierte Experimentierumgebung zur automatischen Spracherkennung vorgestellt. Der realisierte Prototyp beinhaltet bereits die Basisfunktionalitäten zur Verwaltung der Sprachdaten und zur Durchführung von Erkennungsexperimenten. In einem ersten Einsatz wurde die Erkennung der sieben Wochentage verwendet. Ausgehend von den ersten Erfahrungen wurden mögliche Erweiterungen diskutiert.

Literatur

[1] Ian McGraw: *Collecting Speech from Crowds*. In *Crowdsourcing for Speech Processing: Applications to Data Collection, Transcription and Assessment* . John Wiley 2013: 37-71.

[2] Paul Adenot, Chris Wilson: *Web Audio API*. W3C Editor's Draft 06 January 2015

[3] Raphaël Saunier:. *Getting Started with Laravel 4*. Packt Publishing Ltd, 2014.