SPRACHERKENNUNG IM WEB

Konzept-Sheet | Stand: 19. Juni 2009 um 16:49 | Andreas Lutz . hello@andreaslutz.com |

Einführung

Durch meine Diplomarbeit „Audio-visuelle Navigation im Web“ evaluierte ich einen praktikablen Ansatz für eine echte Spracherkennung im Web, wie sie bisher aus dem Offline-Bereich für z. B. Telefonanwendungen oder Diktiersoftware bekannt ist. Bisher ist eine solche Online-Anwendung noch nicht realisiert worden, da technische Gegebenheiten noch nicht ausreichend zur Verfügung standen und keine proprietäre Lösung entwickelt wurde. Daher ist es nicht verwunderlich, dass mein Ansatz eine Symbiose aus verschiedenen Technologien darstellt: Die Flash-Technologie von Adobe, die Java™ Technologie von Sun und eine herkömmliche Web-Server-Infrastruktur. Maßgeblich für die Innovation ist die Entwicklung des Java-Plugins zur aktuellen Version 1.6.

Grundlage

Grundlage für das Konzept ist die Spracherkennungsanwendung Sphinx 4:
„Sphinx-4 is a state-of-the-art speech recognition system written entirely in the JavaTM programming language. It was created via a joint collaboration between the Sphinx group at Carnegie Mellon University, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs (MERL), and Hewlett Packard (HP), with contributions from the University of California at Santa Cruz (UCSC) and the Massachusetts Institute of Technology (MIT).“
Sphinx 4 ist also eine frei verwendbare Spracherkennung mit bereits bestehenden sog. Acoustic-Models für nahezu alle englischen Wörter. Sie ist in sehr vielen Spracherkennungsanwendungen wie Telefonsteuerungen o.ä. implementiert. Aufgrund einer umfangreichen API ist sie auch individuell konfigurierbar und so für fast jeden Zweck einsetzbar.

Über die Sphinx 4 Website stieß ich auf die Anwendung Voce von Tyler Streeter. Voce ist eine Java-Applikation, die bereits eine fertige Mikrofon-Eingabe und ein Verarbeitungsszenario mit Sphinx 4 implementiert. Durch die auf das Wesentliche reduzierte API kann man mit Voce sehr schnell eine eigene Java-Applikation realisieren. So lässt sich z. B. schon mit diesen paar Zeilen eine grundlegende Spracherkennungsanwendung als Java-Applikation erstellen:

```
while (voce.SpeechInterface.getRecognizerQueueSize() > 0)
{
      String s = voce.SpeechInterface.popRecognizedString();
      System.out.println("You said: " + s);
}
```

Somit ist eine echte Spracherkennung schon einmal theoretisch realisierbar. Da die Anwendung aber online ablaufen soll, muss zuerst die Java-Applikation in ein Java-Applet umgeschrieben werden. Der Vorteil eines Java-Applets ist, das es direkt im Browser angezeigt und ausgeführt werden kann, ganz im Gegenteil zur genannten Java-Applikation oder der sog. Java Web Start Technologie. Bei diesen müssen die Anwendungen immer zuerst geladen und dann manuell ausgeführt werden: Keine Alternative also für eine benutzerfreundliche Anwendung. Der Prototyp bildet den Aufbau für eine Online-Spracherkennung ab.

Prototyp

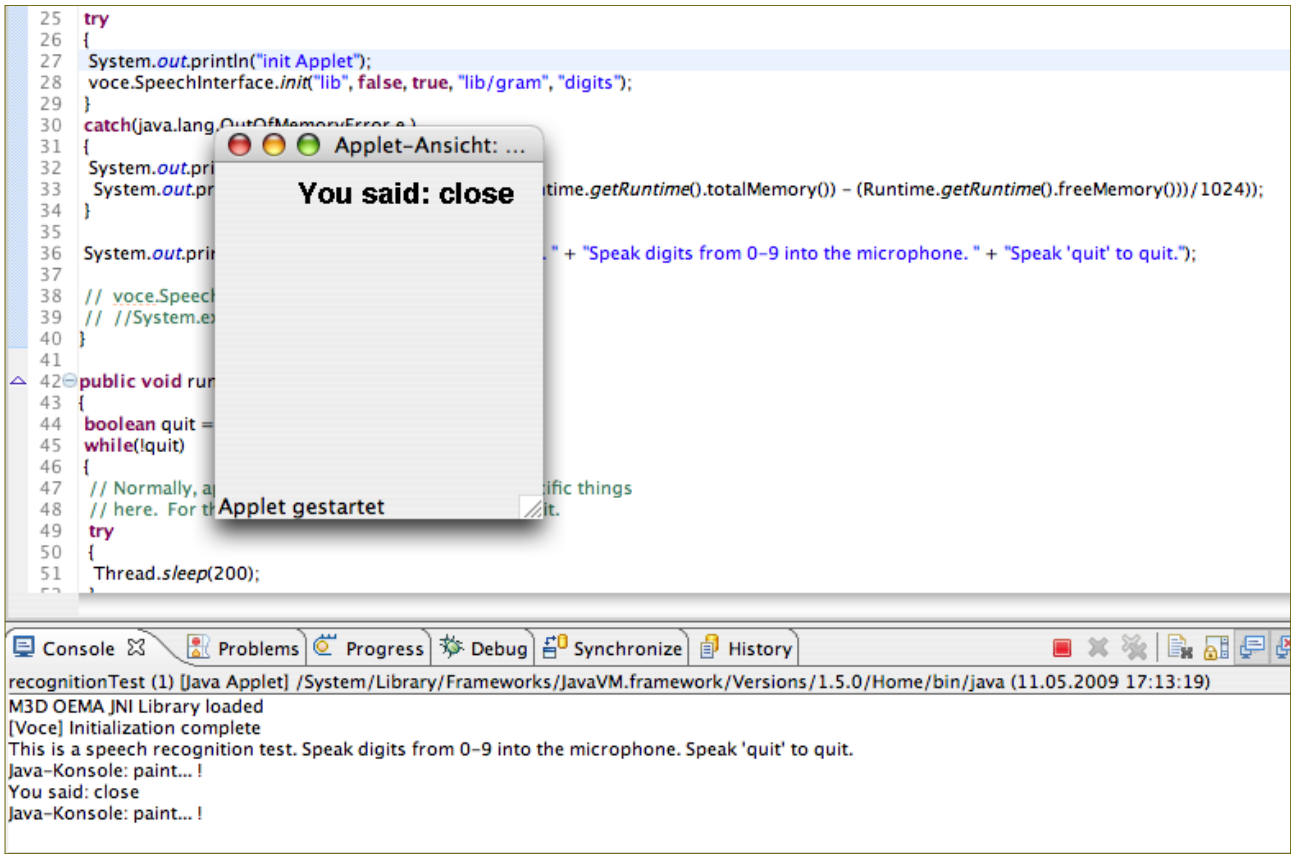

Abb. 1 Spracherkennung im Applet

Wie in Abb. 1 gezeigt, wird meine Spracheingabe „close“ erkannt und als String im Applet ausgegeben. Im nächsten Schritt muss dieser String im Applet zur Laufzeit an Flash weitergeleitet werden. Dies kann man durch einen sog. XML-Socket-Server und die Adressierung an einen bestimmten Port (gewöhnlich 1024 oder höher) realisieren. Diesen Port (der sog. XML-Socket) kann man aus Flash heraus überwachen und auf empfangene XML-Daten überprüfen. Diese XML-Daten, also der erkannte String, kann dann in Flash ausgelesen werden und zur weiteren Befehlsverarbeitung genutzt werden.

Bei diesem Ansatz gibt es zwei Schichten im Browser: Das Applet und Flash. Das Applet übernimmt in dieser Anordnung die alleinige Aufnahme von Mikrofon-Signalen und deren Verarbeitung, d.h. Flash wird hier nur zur Repräsentierung der gewonnenen Daten genutzt. Ein weiterer wichtiger Vorteil in Bezug auf die Performanz. Zudem läuft die ganze Verarbeitung im Applet auch clientseitig ab, was sich wiederum positiv auf Serverperformance und Zugriffszeiten auswirkt. Abb. 2 bildet den Prototypen eines solchen Aufbaus ab. Die Sprach-eingabe „link“ wird erkannt, im Applet (links) verarbeitet und über den XML-Socket-Server an das Flash-File (rechts) weitergeleitet.

Bettet man nun also das Applet in die gleiche HTML-Seite wie die finale Flash-Applikation ein, kann man dieses vorgestellte Prozedere dazu nutzen, eine echte Spracherkennung in Flash zu realisieren. Auffällig beim Prototyp wurde aber schon, dass das Applet nicht mit dem standardmäßig vom System zur Verfügung gestellten Arbeitsspeicher von 64MB einsatzfähig ist, sondern mindestens 128MB Anteil des Speichers des Clients benötigt. Eine solche Erhöhung lässt sich aber durch eine Parameterübergabe (Parameter -Xmx128m) bei der Kompilierung zuverlässig erreichen. Ausserdem ist eine solche künstliche Arbeitsspeichererweiterung manuell in den Java-Einstellungen des Betriebssystems möglich.

Um die genannte Speichererweiterung auch Online zu gewährleisten, kann man per HTML-Parameterübergabe den vom System zur Verfügung gestellten Speicher erweitern:

```
<param name="java_arguments" value="-Xmx128m" />
```

Diese Parameterübergabe ist erst mit der Java-Version 1.6.0 möglich geworden und eröffnet somit u.a. erst die Möglichkeit zur Realisierung einer solchen Anwendung.

Abb. 2 Spracherkennung im Applet via XML-Socket-Server final zu Flash

Status Quo

In der Offline-Testumgebung konnten Versuche erfolgreich abgeschlossen werden. Um eine vollwertige Anwendung für Online zu realisieren bleiben noch folgende Arbeitsschritte: Durch die Sicherheitsbeschränkungen von Applets, die in Browsern ausgeführt werden, ist es derzeit leider nicht möglich, die gleichen Sicherheitsbestimmungen, die für lokal ausgeführte Applets gelten, online anzuwenden. Voce bzw. Sphinx 4 benötigt aber genau einiger dieser Rechte um diese Anwendung auch online korrekt ausführen zu können. Ein weitere Punkt der noch weiterentwickelt werden müsste, ist die Größe der Acoustic Models von Sphinx 4. Diese sind mit ca. 15MB sehr groß und müssen initial geladen werden. Doch durch Beschränkung auf bestimmte Wörter, Zahlen u.ä. ist dieses Paket reduzierbar.

Szenarien

Ein theoretisches Anwendungsszenario mit Spracherkennung wäre die Navigation einer Website mit Wörtern oder etwa eine vollwertige eLearning-Umgebung für Online. Im Umfeld des eLearnings wäre es unter diesen Voraussetzungen möglich, vom User gesprochene Wörter oder Sätze als Zeichenketten umzusetzen und so in der Applikation weiter zu verarbeiten. Dabei sind Anwendungen wie interaktive Vokabel-Tests, interaktive Multiple-Choice-Tests oder direkte Kommunikation mit dem Dozent via Voice-Chat möglich.Eine weitere Einsatzmöglichkeit wäre die aus dem Offline-Bereich bekannte Anwendung einer Diktiersoftware für z. B. Microsoft Word. Ein Online-Äquivalent wäre unter Einsatz der Spracherkennungsapplikation mit Adobe Buzzword oder Google Docs möglich.