Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Technical Memo
TM-05-01

Portierung von Merkmalsextraktion auf die PocketPC-Plattform

Michael Feld

April 2006

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210
E-Mail: info@dfki.uni-kl.de

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341
E-Mail: info@dfki.de

WWW: http://www.dfki.de

Deutsches Forschungszentrum für Künstliche Intelligenz

DFKI GmbH

German Research Center for Artificial Intelligence

Founded in 1988, DFKI today is one of the largest nonprofit contract research institutes in the field of innovative software technology based on Artificial Intelligence (AI) methods. DFKI is focusing on the complete cycle of innovation — from world-class basic research and technology development through leading-edge demonstrators and prototypes to product functions and commercialization.

Based in Kaiserslautern and Saarbrücken, the German Research Center for Artificial Intelligence ranks among the important “Centers of Excellence” worldwide.

An important element of DFKI’s mission is to move innovations as quickly as possible from the lab into the marketplace. Only by maintaining research projects at the forefront of science can DFKI have the strength to meet its technology transfer goals.

The key directors of DFKI are Prof. Wolfgang Wahlster (CEO) and Dr. Walter Olthoff (CFO).

DFKI’s six research departments are directed by internationally recognized research scientists:

- Image Understanding and Pattern Recognition (Director: Prof. Thomas Breuel)
- Knowledge Management (Director: Prof. A. Dengel)
- Intelligent Visualization and Simulation Systems (Director: Prof. H. Hagen)
- Deduction and Multiagent Systems (Director: Prof. J. Siekmann)
- Language Technology (Director: Prof. H. Uszkoreit)
- Intelligent User Interfaces (Director: Prof. W. Wahlster)

Furthermore, since 2002 the Institute for Information Systems (IWi) (Director: Prof. August-Wilhelm Scheer) is part of the DFKI.

In this series, DFKI publishes research reports, technical memos, documents (eg. workshop proceedings), and final project reports. The aim is to make new results, ideas, and software available as quickly as possible.

Prof. Wolfgang Wahlster
Director

Portierung von Merkmalsextraktion auf die PocketPC-Plattform

Michael Feld

DFKI-TM-05-01

This work has been supported by a grant from The Federal Ministry of Education, Science, Research, and Technology (FKZ ITW-01 IN C02).

ISSN 0946-0071

Portierung von Merkmalsextraktion auf die PocketPC-Plattform

Michael Feld

28. August 2005

Zusammenfassung

Diese Arbeit stellt eine Softwarelösung zur Extraktion von phonetischen Merkmalen, die als Grundlage für andere Applikationen dienen, aus einem digitalen Sprachsignal auf der PocketPC-Plattform vor. Die Entwicklung der Lösung wird schrittweise von der Festlegung der Kriterien über den Entwurf bis hin zur Implementierung beschrieben. Das Ergebnis ist eine fertige Bibliothek, welche die geforderten Merkmale bietet. Zuvor wird ein kurzer Überblick über das Projekt M3I, in dessen Rahmen diese Arbeit entstand, sowie die computergestützte Sprachanalyse im Allgemeinen gegeben. Darüber hinaus werden einige softwaretechnische Aspekte der Applikationsportierung angesprochen und im Hinblick auf die mobile Plattform konkretisiert.

1 Zweck der Merkmalsextraktion

Sprachmerkmalsextraktion auf akustischer Ebene ist der Ausgangspunkt für viele Algorithmen und Verfahren, welche zur wissenschaftlichen Sprachuntersuchung und -analyse sowie in kommerziellen Anwendungen eingesetzt werden. Diese Arbeit ist im Rahmen von M3I[1] entstanden, einem am DFKI[2] durchgeführten Projekt zur Entwicklung von Verfahren zur multimodalen Interaktion mit mobilen Geräten. Ziel des Teilprojekts AGENDER[3] ist es hierbei, den Benutzer eines mobilen Gerätes sowie den aktuellen Kontext zu klassifizieren, ohne den Benutzer zur Eingabe dieser Informationen aufzufordern. Als Quelle für diese Datenakquisition wurde die Spracheingabe des Benutzers bestimmt, welche nebenläufig im normalen Betrieb ausgewertet wird. Auf Basis einer solchen Sprachprobe ist es dann möglich, Rückschlüsse auf das Alter und Geschlecht des Benutzers zu ziehen, sowie über die Geräuschkulisse der Umgebung. Gegenwärtig werden für alle diese Eigenschaften jeweils zwei Kategorien unterschieden, z.B. jung und alt für die Altersklassifikation. Diese Information kann dann wiederum anderen Anwendungen zur Verfügung gestellt werden, wobei die möglichen Einsatzbereiche sehr vielfältig sind. Das Alter kann von Computeranwendungen beispielsweise generell gut zur Adaption der Lesbarkeit für ältere Menschen genutzt werden.
Die Architektur des vollständigen Systems gestaltet sich wie folgt: Das mobile Gerät repräsentiert aus Sicht von M3I den Client. Auf dem Client wird eine Applikation ausgeführt, die vom Endanwender gestartet wurde, z.B. ein digitaler Einkaufsassistent. Wenn diese Applikation die AGENDER-Funktionen in Anspruch nehmen möchte, so kann sie auf die Funktionen der M3I-Bibliothek zugreifen, welche als DLL[4] für den PocketPC implementiert wurde. Die Anwendung kann dann eingehende Sprachproben des Benutzers (z.B. für Sprachsteuerung der Anwendung) an die Bibliothek weiterleiten, und erfährt auf Nachfrage die aktuelle Einschätzung des Benutzers bzgl. Alter und Geschlecht. Bisher wurde die Verarbeitung der Sprachdaten ausschließlich auf einem externen Server durchgeführt. Dieser M3I-Server kann Verbindungsanforderungen mehrerer anonymer mobiler Clients annehmen, z.B. über ein Funknetzwerk (WLAN), und führt auf Anfrage hin die Klassifizierung durch, wobei hier

[1] A Mobile, Multi-modal and Modular Interface, `https://xantippe.cs.uni-sb.de/~cmueller/m3i/php/website2.php`
[2] Deutsches Forschungszentrum für Künstliche Intelligenz
[3] Age+Gender
[4] Dynamic Link Library, siehe auch Abschnitt 9

– je nach Volumen – ein Cluster zum Einsatz kommen kann. Eingehendere Informationen zur M3I-Architektur sind in [5] nachzulesen.
Ein zukünftiges Ziel ist es, diese Klassifizierung auch durchführen zu können, wenn die Verbindung zum Server eingeschränkt ist, oder sogar wenn gar kein Server vorhanden ist. Denkbar sind dann auch verschiedene Szenarien paralleler oder kooperativer Verarbeitung.
Die primären Klassifizierungsmethoden von AGENDER sind künstliche neuronale Netze, die jeweils für Alter und Geschlecht existieren. Neuronale Netze können unterschiedliche Strukturen aufweisen; typischerweise bestehen sie aus Knoten mit Formeln, welche jeweils ihre Eingabewerte unter einer bestimmten Gewichtung kombinieren und das Ergebnis mit einem vordefinierten Schwellenwert vergleichen und so zu einem binären Ergebnis kommen. Das Netz entsteht dadurch, dass mehrere solcher Knoten miteinander verbunden werden. Als Eingabewerte für diese Klassifizierer werden die akustischen Sprachmerkmale verwendet, die aus der Eingabedatei extrahiert werden. Dies führt in der servergestützten Architektur ein Prozess durch, der auf dem Server selbst oder einem lokalen Cluster läuft. Ist jedoch der Server nicht verfügbar, so würde diese Aufgabe dem Client zufallen.
AGENDER ist das für diese Arbeit ausschlaggebende Einsatzgebiet der Merkmalsextraktion, jedoch sind auch noch zahlreiche weitere Anwendungen denkbar, die eine Merkmalsextraktion auf dem PocketPC nutzen könnten, z.B. zur wissenschaftlichen Analyse von beliebigen Audiodaten oder um – analog zu AGENDER – weitere Informationen aus der Sprachprobe zu entnehmen. Für PC und Macintosh gibt es zu diesem Zweck die Softwaresuite PRAAT[5], die über Skripte in andere Anwendungen eingebunden werden kann.

2 Anforderungen

Die Software zur Merkmalsextraktion, die für diese Arbeit entwickelt wurde, soll neben der eigentlichen Werteberechnung noch einige Rahmenanforderungen erfüllen, um erfolgreich mit der bereits vorhandenen Architektur eingesetzt werden zu können.
Als erstes muss die Ausführung unter *Intel StrongARM* und kompatiblen Prozessoren gewährleistet sein, da diese die Mehrzahl der eingesetzten Geräte ausmachen. Als Entwicklungsgerät diente der *HP Jornada 568*, welcher mit einem *ARM SA1110* Prozessor ausgestattet ist. Von Seiten des Betriebssystems wird Kompatibilität mit *Windows CE 3.0* und höher (*PocketPC 2002 / Windows Mobile 2003*) vorausgesetzt, welche ebenfalls einen de-facto Standard für PocketPCs darstellen. Diese beiden Faktoren bewirken bereits eine starke Einschränkung in Bezug auf die Auswahl der Entwicklungstools. Einziger echter Kandidat sind die (kostenlos) von Microsoft angebotenen Werkzeuge *eMbedded Visual C++ 4.0* (kurz: *eVC++*) und *PocketPC 2002 SDK*. Dabei stellt *eVC++* eine integrierte Entwicklungsumgebung mit eigenem Compiler dar, die stark an das verwandte *Visual C++* erinnert, jedoch für mobile Geräte optimiert wurde. Für den Compiler müssen einige benutzerdefinierte Optionen angegeben werden, um die Kompilierung für *PocketPC 2002* zu ermöglichen; herstellerseitig wird nur *PocketPC 2003* und höher unterstützt. Das *PocketPC 2002 SDK* enthält plattformspezifische Dateien für *Windows CE 3.0*, wie z.B. Header-Dateien und Bibliotheken, welche zum Kompilieren benötigt werden, sowie umfangreiche Dokumentation und eine Emulator-Software. Obwohl die Entwicklungsumgebung an sich gut geeignet ist für die PocketPC-Entwicklung und auch keine gravierenden Schwächen aufweist, so ist das Projekt mit dieser Entscheidung aber festgelegt in Hinsicht auf den existierenden Quellcode, der direkt in das Programm übernommen werden kann. Die Alternative, PocketPCs mit *Linux* als Betriebssystem auszuwählen, hätte in Bezug auf die Entwicklungsmöglichkeiten sicher eine größere Flexibilität bedeutet, beispielsweise was die Auswahl des Compilers betrifft. Allerdings wäre dadurch die Integration mit bestehenden Anwendungen erschwert worden, denn gegenwärtig sind die große Mehrzahl der Softwarelösungen für *Windows CE* konzipiert.
Eine weitere Anforderung an die Software ist, dass sie möglichst universell in andere Programme integriert werden kann, d.h. sie soll die Extraktionsmethoden in Form einer Bibliothek zur Verfügung stellen. Für die exakte Implementierung gibt es dann noch mehrere Möglichkeiten, auf die in Abschnitt 9 näher eingegangen wird.

[5] `http://www.praat.org`

3 Sprachverarbeitung mit dem Computer

Die Merkmalsextraktion ist eine Form der Sprachverarbeitung. Diese erfolgt auf dem Computer stets digital, d.h. die Sprachinformation liegt in Form einer binären Audiodatei vor oder wird – sofern eine Aufnahme in Echtzeit verarbeitet wird – als solche temporär im Arbeitsspeicher angelegt. Diese binäre Repräsentation enthält häufig weniger Information als die ursprüngliche analoge Sprachprobe (siehe [7]), da bei der Digitalisierung Informationen verloren gehen können. Deshalb spielt die Qualität der Aufnahme eine große Rolle und wirkt sich auch auf die Ergebnisse der Merkmalsextraktion aus. So können Abweichungen in den berechneten Werten häufig auf verschiedene Formate der Audiodateien zurückgeführt werden. Aus diesem Grund kann es empfehlenswert sein, die Daten in einem zuvor festgelegten gemeinsamen Format abzuspeichern oder sie zumindest vor der Analyse in ein gemeinsames Format zu überführen (Konvertierung).
Die Kriterien, welche die Qualität der Audiodaten bestimmen, sind *Samplingrate*, *Bitrate* und *Anzahl der Kanäle*. In der Natur sind akustische Phänomene Schallwellen. Um sie in eine digitale Form zu bringen, müssen diese Wellen durch Abtastpunkte (*Samples*) approximiert werden (*Quantisierung*). Dabei können einfache Speicherverfahren oder aufwändige Komprimierungsalgorithmen wie [8] zum Einsatz kommen. Die Anzahl der aufgezeichneten Samples pro Sekunde stellt die Samplingrate dar, welche in Hertz angegeben wird (44.100 Hz entspricht CD-Qualität). Die Samplingrate bestimmt auch die maximale Frequenz, die aufgezeichnet werden kann, wobei nach dem in [7] vorgestellten *Sampling-Theorem* die Samplingrate mehr als das doppelte der höchsten im Signal auftretenden Frequenz (der *Nyquist-Frequenz*) betragen soll. Die menschliche Stimme deckt bei gesprochenen Äußerungen nach [2] und [3] etwa einen Bereich von 75 bis 500 Hz ab, so dass mit 16 kHz eine ausreichende Auflösung erreicht wird. Die Bitrate gibt an, mit welcher Genauigkeit ein einzelnes Sample gespeichert werden kann (z.B. 16 Bit bei CDs). Eine höhere Bitrate ergibt eine breitere Klangdynamik. Mehrere Kanäle werden verwendet, um Raumklang zu simulieren. Jeder Kanal kann eigene Audiodaten enthalten und die Kanäle können prinzipiell wie verschiedene Aufnahmen behandelt werden.
In Hinblick auf Qualität sollten die Samplingrate und Bitrate so hoch wie möglich gewählt werden, da dies genauere Ergebnisse liefert. In der Praxis gelten hierfür jedoch einige Beschränkungen:

1. Die Hardware (insb. das Mikrofon) und der Aufnahmetreiber müssen die geforderten Werte unterstützen. Gerade was die Bitrate angeht wird dies sehr oft durch die Hardware auf 16 Bit begrenzt. Auch sollte man bei mobilen Geräten davon ausgehen, keine besonders hochwertigen Mikrofone vorzufinden. Selbst die OEM[6]-Treiberimplementierung ist oftmals nur minimal vorhanden und fehleranfällig.

2. Je höher Samplingrate und Bitrate gewählt werden, desto größer ist der anfallende Speicherverbrauch. Wird die Sprachprobe nach der Verarbeitung wieder gelöscht, so ist das zwar nur ein unbedeutender Punkt, aber wenn die Sprachdateien archiviert werden sollen, so wird der Speicherverbrauch ein gewichtiges Argument sein. Nähere Untersuchungen zu diesem Thema sind in [7] zu finden.

3. Mit höherer Aufnahmequalität steigt auch die zu verarbeitende Datenmenge und damit die benötigte Rechenzeit. Dieser Punkt ist besonders wichtig bei Echtzeitanwendungen auf mobilen Geräten, da deren Leistungsfähigkeit immer noch sehr gering ist verglichen mit Desktop-PCs.

4. Wie zuvor erwähnt ist es sinnvoll, die Audiodaten vor der Merkmalsextraktion und Analyse in ein gemeinsames Format zu bringen, um den Vergleichswert der erhaltenen Werte zu erhöhen. Bei der Wahl dieses gemeinsamen Formates sollte daher das jeweilige Minimum der anzunehmenden Eingabeformate bzgl. Samplingrate und Bitrate für unterschiedliche Geräte und Applikationen gewählt werden, da Formate mit niedrigeren Raten nicht (oder nur unzulänglich) in ein Format mit größeren Raten konvertiert werden können.

[6] Original Equipment Manufacturer

Für die Sprachmerkmalsextraktion genügt ein einziger Audiokanal. Sollte die Aufnahme im Stereo-Format vorliegen, so können die Kanäle vor der Merkmalsextraktion gemischt werden, oder es wird ein Kanal ausgewählt.

4 Verwendete Sprachmerkmale

Aus einer digitalisierten Sprachprobe können mit Hilfe von Algorithmen eine ganze Reihe von Merkmalen extrahiert werden. Die Softwaresuite PRAAT wurde genau für diesen Zweck entwickelt, und die Online-Hilfe zum Programm gibt einen kleinen Einblick in die vielfältigen Möglichkeiten, Sprachsignale mittels Algorithmen zu untersuchen.
In dieser Arbeit soll jedoch nur auf die Merkmale eingegangen werden, welche für die spätere Klassifizierung von Alter und Geschlecht durch die neuronalen Netze relevant sind und deren Bedeutung in [6] näher beschrieben wird. Dabei handelt es sich um die folgenden drei Gruppen von Merkmalen:

Pitch

Das Merkmal *Pitch* beschreibt die Tonlage eines Sprachsignals. Diese ergibt sich aus der Periodendauer bzw. Frequenz der Schallwellen, welche in der Natur kontinuierlich sind, und kann daher nur schwer an diskreten Stellen erfasst werden. Zur Bestimmung der Tonlage an einem bestimmten Punkt muss immer auch ein Zeitfenster links und rechts dieses Punktes betrachtet werden, um eine Periodizität feststellen zu können. Pitch wird in Hertz, Mel oder Halbtönen angegeben. Aus einer endlichen Menge von Pitch-Werten für ein Sprachsignal kann durch parabolische Interpolation eine Funktion der Zeit erstellt werden, die als *Pitch-Kontur* des Signals bezeichnet wird. Für die Klassifizierung durch AGENDER sind folgende Eigenschaften der Funktion von Interesse:

- Höchste auftretende Frequenz (Maximum)
- Niedrigste auftretende Frequenz (Minimum)
- Durchschnittliche Tonlage
- Standardabweichung
- 50%-Quantil (Median)
- Absolute mittlere Steigung

Jitter

Durch *Jitter* können Abweichungen des Pitch-Wertes beschrieben werden, wobei jeweils unterschiedlich große Umgebungen betrachtet werden können. Voraussetzung ist, dass das Signal bereits in Perioden eingeteilt wurde. Der Jitter wird dann berechnet, indem der Betrag der Frequenzdifferenzen aller Paare von aufeinander folgenden Perioden bestimmt und aus diesen dann der Mittelwert gebildet wird. Dieser wird dann üblicherweise prozentual zur durchschnittlichen Frequenz des Signals angegeben. Folgende Variationen der Jitter-Funktion werden unterschieden:

- Lokaler *Jitter* (absolut)
- Lokaler *Jitter*: Jitter im Verhältnis zur durchschnittlichen Frequenz
- *Relative Average Perturbation (RAP)*: Hierbei wird die Abweichung einer Periode vom Mittelwert aus der Periode selbst und den beiden benachbarten Perioden betrachtet.
- *Five-point Period Perturbation Quotient (PPQ5)*: Differenz aus der Periode und dem Mittelwert aus ihr selbst und den vier nächsten Nachbarn
- *Difference of Differences of Periods (DDP)*: Wählt von der Abweichungen zweier aufeinander folgender Perioden nochmals den Betrag der Differenz.

Shimmer

Der Parameter *Shimmer* ist sehr eng mit dem Jitter verwandt. Es handelt sich um das Gegenstück zur Frequenz im Bereich der Amplitude. Hier werden Schwankungen der Amplitude innerhalb verschieden großer benachbarter Umgebungen betrachtet und üblicherweise auch wieder zur durchschnittlichen Amplitude des Audiosignals in Bezug gesetzt. Die einzelnen relevanten Funktionen sind:

- Lokaler *Shimmer*: Mittelwert aus dem Betrag der Differenz der Amplituden in je zwei aufeinander folgenden Perioden im Verhältnis zur durchschnittlichen Amplitude
- Absoluter lokaler *Shimmer*: Mittelwert aus dem Betrag des Zehnerlogarithmus der Differenz der Amplituden in je zwei aufeinander folgenden Perioden. Das Ergebnis wird mit 20 multipliziert.
- *Three-point Amplitude Perturbation Quotient (APQ3)*: Analog zum Jitter RAP wird als Subtrahend der Mittelwert aus der Periode und ihren beiden Nachbarn gebildet.
- Analog *APQ5* und *APQ11* mit den vier bzw. zehn nächsten Nachbarn
- *Difference of Differences of Periods (DDP)*: Wählt von der Abweichungen zweier aufeinander folgender Perioden den Betrag der Differenz der Amplituden.

Um die genannten Merkmale zu erhalten, geht der M3I-Server wie folgt vor: Zuerst wird ein eingehendes digitales Sprachsignal durch Up- bzw. Downsampling in ein fest definiertes Format gebracht. Die Ausgabedatei wird dann an ein PRAAT-Skript übergeben, welches automatisch alle oben aufgeführten Sprachmerkmale extrahiert. Im nächsten Abschnitt soll erläutert werden, wie dieser Schritt auf dem PocketPC nachvollzogen wurde.

5 Gründe für die Portierung von PRAAT

Die wichtigste Entscheidung, die zu Beginn des Projekts getroffen werden musste, war die Art und Weise, auf die die Extraktionsalgorithmen implementiert werden sollten. Hier kamen prinzipiell zwei Möglichkeiten in Frage: Verwendung eines bereits vorhandenen Werkzeugs oder vollständige Neuimplementierung aller benötigten Algorithmen. Im Falle dass auf ein bereits existierendes Programm oder frei verfügbaren Quellcode zurückgegriffen würde, müssten noch diverse Fragen zur Integration erörtert werden, und hierbei handelt es sich um einen gewichtigen Punkt, der über Erfolg oder Scheitern einer Methode entscheiden kann. Die typischen Integrationsverfahren sind im Einzelnen:

Einbindung einer Bibliothek: Im Optimalfall werden vorhandene Algorithmen über eine öffentliche Schnittstelle in Form einer Bibliothek (z.B. DLL) zur Verfügung gestellt. Wenn solch eine Bibliothek existiert und alle gewünschten Funktionen enthält, so kann diese meist direkt ohne großen Aufwand in das Rahmenprojekt integriert werden. Dies setzt natürlich voraus, dass die Bibliothek, da sie kompilierten, ausführbaren Code enthält, auf der Zielplattform lauffähig ist. Auch muss die Sprache des Rahmenprojekts (im Falle des M3I-Clients C++) Aufrufe aus DLLs unterstützen.

Einbindung einer Programmdatei: Steht für das gewünschte Programm keine Bibliothek zur Verfügung, sondern nur eine ausführbare Programmdatei, so kann diese möglicherweise ebenfalls genutzt werden. Dazu muss das Programm Parameter wie die zu verarbeitende Sprachdatei aus der Umgebung einlesen können (z.B. über die Kommandozeile, eine Umgebungsvariable oder eine Konfigurationsdatei), die Ergebnisse in programmatisch verwertbarer Form zurückliefern (z.B. über die Konsolenausgabe oder eine temporäre Datei) und den Vorgang vollautomatisch ohne GUI[7] und Benutzereingriffe durchführen können. Wie bei der Bibliothekseinbindung muss auch hier die Zielplattform unterstützt

[7] Graphical User Interface, grafische Benutzeroberfläche

werden, allerdings muss keine Schnittstelle zwischen den Programmiersprachen bestehen, da diese bereits durch den Aufruf der ausführbaren Datei über das Betriebssystem vorgegeben ist.

Diese Lösung hat immer gewisse Geschwindigkeitsnachteile gegenüber einer Bibliothek, da der Startvorgang des Programms zusätzliche Zeit im Betriebssystemkernel benötigt.

Portierung eines Programms: Wenn ein Programm für die benötigte Plattform nicht existiert, so kann es als Bibliothek oder ausführbare Datei nicht eingebunden werden. Falls der Quellcode des Programms erhältlich ist, so besteht jedoch die Möglichkeit einer Portierung auf die Zielplattform.

Hierbei muss vor allem der vorhandene Code an die API[8] des Zielbetriebssystems und -prozessors angepasst werden und evtl. compilerspezifisch verändert werden. Je nach Unterschiedlichkeit der Plattformen kann sich dies mehr oder weniger umfangreich gestalten. Beispielsweise gibt es einige Prozessoren, welche keine Fließkommaoperationen unterstützen, sodass diese dann bei einer Portierung ggf. komplett ersetzt werden müssten. Bei Sprachen, die in einer virtuellen Maschine ausgeführt werden (z.B. *Java*, *.NET*) sind im Allgemeinen keine Änderungen dieser Art notwendig, da sie speziell auf Plattformunabhängingkeit ausgelegt sind. Weiterhin müssen bei einer Portierung plattformspezifische Besonderheiten auf Ressourcenebene beachtet werden, die meist nicht durch den Compiler entdeckt werden. So hat der PocketPC beispielsweise eine kleinere Anzeige als ein Desktop-Computer, und alle Anwendungen sollten auf diesen Umstand hin optimiert werden.

PRAAT ist eine frei verfügbare Anwendung zur Sprachanalyse und -synthese, die 1992 von Paul Boersma und David Weenink vom *Department of Phonetics* an der Universität von Amsterdam geschrieben wurde und seitdem ständig weiterentwickelt wird. Sie beinhaltet umfangreiche Funktionen und Algorithmen zur Arbeit mit Sprachdateien, sowie Methoden zur Visualisierung der Ergebnisse. PRAAT erlaubt auch das Schreiben eigener Skriptdateien, um neue Funktionen zu definieren oder Sprachproben automatisiert bzw. stapelweise abzuarbeiten, was es für die Integration in andere Projekte qualifiziert.
Verweise in der Literatur und Recherchen im Internet zum Thema „computergestützte Sprachanalyse“ belegen die weite Verbreitung von PRAAT. Die Tatsache, dass es – einschließlich Quellcode – frei verfügbar ist (veröffentlicht unter der GNU[9] Lizenz), unterstützt seine Popularität. Auf der anderen Seite ist PRAAT aber auch eine von relativ wenigen professionellen Lösungen, die in diesem Bereich angeboten werden. Die genannten Gründe sowie die Tatsache, dass es alle benötigten Algorithmen beherrscht, waren ausschlaggebend für den Einsatz im M3I-Server.
Da PRAAT schon einige Zeit erfolgreich im M3I-Server eingesetzt wurde, lag es nahe, das Phonetikwerkzeug auch auf dem PocketPC zu verwenden. Das war allerdings nicht ohne weiteres möglich, da PRAAT von Haus aus nur als Version für Windows auf dem Desktop-PC, Linux und Macintosh existiert. Außerdem beeinflusst die Integration über Skriptdateien auf dem mobilen Gerät die Geschwindigkeit wesentlich stärker und ist nicht zweckmäßig.
Eine Portierung von PRAAT auf den PocketPC und Umstrukturierung als Library war durch den Quellcode in C prinzipiell möglich. Eine erste Analyse des Quellcodes ergab, dass eine vollständige Portierung mit einem gewissen Aufwand verbunden war und daher auch weitere Alternativen in Betracht gezogen werden sollten. Da die Suche nach weiteren geeigneten Werkzeugen im Wesentlichen erfolglos verlaufen war, kam als Alternative nur eine Neuimplementierung der Algorithmen in Frage.
Die Idee einer eigenen Implementierung konnte sich aus mehreren Gründen nicht durchsetzen. Zum einen muss zum Erhalt der Sprachmerkmale Jitter und Shimmer eine Periodifizierung (Periodenerkennung) der Spracheingabe durchgeführt werden, ein nicht triviales Verfahren, das den Aufwand der Neuimplementierung schnell mit dem der Portierung aufwiegt. Für PRAAT spricht darüber hinaus die überragend gute Qualität, die auf jahrelanger Forschung

[8] Application Programming Interface
[9] `http://www.gnu.org/licenses/gpl.html`