

Dynamische Satzgenerierung und Sprachausgabe für einen mobilen Serviceroboter

Christopher Parlitz, Bernd Amann und Martin Hägele

Fraunhofer IPA, Institut für Produktionstechnik und Automatisierung,
Nobelstraße 12, 70569 Stuttgart

Zusammenfassung. Die vorliegende Arbeit befasst sich mit der Entwicklung einer Sprachausgabe für ein mobiles Robotersystem. Dazu wurde zum einen ein auf die Anwendung zugeschnittenes Text-To-Speech Verfahren und zum anderen eine Grammatik, die eine dynamische Satzgenerierung ermöglicht, implementiert. Die Verfahren arbeiten unabhängig voneinander und können daher auch gegen andere Komponenten ausgetauscht werden. Die Programme wurden als Client-Server Applikationen programmiert, um möglichst flexibel die Sprachausgabe einsetzen zu können.

1 Einleitung

Obwohl Sprache nur ein kleiner Teil der menschlichen Interaktion ist, wird eine zuverlässige und qualitativ hochwertige Sprachein- und -ausgabe bei einem Serviceroboter mit fortschreitender Entwicklung immer interessanter. Die natürlichsprachliche Kommunikation hat einige Vorteile. Zuerst einmal ist kein besonderes Training (des Benutzers) nötig, wie das beispielsweise bei einer Tastatur mit Bildschirm nötig wäre. Außerdem hat der Benutzer Augen und Hände frei und ist nicht an einen Ort gebunden.

Das Projekt wurde in zwei Teile aufgeteilt. Zuerst wurde eine Sprachausgabe entwickelt, anschließend ein Modul zur Satzgenerierung. Beide Module haben dokumentierte Schnittstellen und sind somit mit geringem Aufwand in eine Systemarchitektur integrierbar.

2 Sprachausgabe

In diesem Teil der Arbeit wurde eine Sprachausgabe für einen mobilen Serviceroboter entwickelt. Für die eigentliche „Text-to-Speech“-Synthese wurde ein „Unit selection“-Verfahren unter Verwendung des Sprachsynthesepakets Festival [1] genutzt. Ein „Text-to-Speech“-System (kurz: TTS) wird oft auch als Vorleseautomat bezeichnet. Die Eingabe ist eine Äußerung im Klartext, die (gesprochene) Ausgabe kann über Lautsprecher oder direkt in eine Audiodatei (z. B. *.wav) erfolgen. Das bei diesem Projekt verwendete Sprachsyntheseverfahren gehört zur Gruppe der konkatenativen Verfahren.

2.1 „Unit selection“

Die Idee der konkatenativen Synthese ist, aufgenommene natürlichsprachliche Äußerungen miteinander zu verketteten und so zu einer natürlich klingenden Sprachausgabe zu kommen. Eines der wichtigsten Problemstellungen bei dieser Methode ist das Finden der sinnvollsten Einheitenlänge. Man muss in der Regel zwischen längeren und kürzeren Einheiten abwägen. Längere Einheiten erfordern wesentlich mehr Speicher und eine höhere Anzahl von Einheiten, haben dafür aber weniger Verkettungsstellen und eine natürlicher klingende Ausgabequalität. Kürzere Einheiten benötigen weniger Speicher, der Aufwand bei der Erzeugung der Stimme (Alignierung) ist allerdings höher.

Die meisten konkatenativen Synthesemethoden nutzen akustische Einheiten mit einer festen Länge (z. B. Diphone). Im Unterschied dazu hat die „Unit selection“-Synthese einen anderen Ansatz: die Länge der Einheiten ist variabel von einem Phonem bis hin zu einem ganzen Satz, was diese Methode sehr flexibel macht und die Qualität der Sprachausgabe in einem begrenzten Korpus erhöht. Im Durchschnitt ist die Einheitenlänge bei der „Unit selection“ höher als die Länge von Diphonen oder Halbsilben (nach [2]). Der Auswahlalgorithmus findet bei einer begrenzten Domäne meistens sogar ganze Wörter als Einheiten, was die Anzahl der Verkettungsstellen gering hält und gut für die Sprachqualität ist. Im idealen Fall wird die gesamte Äußerung als Einheit gefunden und die Sprachausgabe spielt diese lediglich ab.

Allerdings muss wie bei grundsätzlich jeder konkatenativen Synthesemethode der Sprachkorpus zuerst erzeugt werden, damit eine individuelle Stimme entsteht und auch der Korpus die zu sprechenden Sätze gut abbildet.

2.2 Algorithmen

Das in Festival [1] verwendete „Unit selection“-Verfahren wurde von Alan W. Black und Paul Taylor 1997 erstmals publiziert [3]. Es verbindet das Verfahren der kontextuellen Klassifikation (Cluster-Algorithmus, z. B. [4]) mit dem Einheiten-Auswahlverfahren nach Hunt und Black [5].

Bei der kontextuellen Klassifikation wird eine Menge von Merkmalen in Äquivalenzklassen (Cluster) eingeteilt. Die Einteilung erfolgt mit Hilfe von binären Entscheidungsbäumen. Die Knoten von Entscheidungsbäumen stellen kontextuelle Fragen dar, anhand derer die Klassifikation erfolgt. Das gesamte Klassifikationsverfahren wird im Voraus durchgeführt, dadurch muss während der Synthese nur noch die strukturierte Datenbank durchsucht werden, was erheblich Rechenzeit einspart. Um den Abstand zwischen Einheiten des gleichen Phonemtyps zu ermitteln, wird ein akustisches Distanzmaß verwendet. Realisiert wird das durch verschiedene akustische Vektoren. Mit dem akustischen Distanzmaß werden Mengen (Cluster) von Einheiten so erzeugt, dass die akustische Distanz („impurity“) der Einheiten innerhalb eines Clusters minimal ist.

Das Einheiten-Auswahlverfahren verwendet zwei Arten von Kosten, um die bestmöglichen Einheiten auszuwählen. Die Zielkosten und die Konkatenationskosten sollten beide jeweils minimiert werden. Je geringer die Zielkosten, desto

besser passt eine Einheit aus der Datenbank auf das Anforderungsprofil. Wie gut sich aufeinanderfolgende Einheiten der Datenbank an der Konkatenationsstelle zusammenfügen lassen, wird durch die Konkatenationskosten dargestellt. Je höher die Kosten, desto schwieriger ist die Anpassung der Einheiten an der Verkettungsstelle. Da die zur Verfügung stehenden Einheiten meist nur geringe Unterschiede in den Eigenschaften aufweisen, werden die Konkatenationskosten durch einen Gewichtungsfaktor etwas höher gewichtet.

2.3 Umsetzung

Nach einer Analyse der Einsatzszenarien des Serviceroboters „Care-O-bot 3“ wurde der Sprachkorpus von einem professionellen Sprecher erzeugt. Bei der Definition des Korpus wurde darauf geachtet, dass alle 48 Phoneme des Deutschen enthalten sind. Damit können, obwohl der Korpus nur eine begrenzte Domäne abdeckt und lediglich 379 verschiedene, vom Roboter häufig gebrauchte Wörter umfasst, auch unbekannte Wörter synthetisiert werden.

Um zur einsetzbaren Stimme zu gelangen, wurden die Sprachdaten einer qualitativen Kontrolle unterzogen und danach aligniert. Mit Hilfe eines Aligners wurden die Audiodateien grob in drei Spuren aufgeteilt (Phonem, Silbe, Wort). Eine aufwändige manuelle Überprüfung ist auf Grund der mangelnden Genauigkeit und Fehlerfreiheit im Anschluss nötig.

Wenn die Sätze nur von Wörtern, die im Korpus enthalten sind, erzeugt werden, ist die Sprachqualität sehr hoch. Es treten gelegentliche minimale Schwankungen der Sprechgeschwindigkeit und kleinere Probleme mit der Prosodie (z. B. bei Fragesätzen) auf. Aufgrund des kleinen Sprachkorpus und der konkatenativen Synthesemethode sind Probleme dieser Art nicht völlig zu verhindern.

Die Qualität der Synthese fällt bei unbekanntem Wörtern teilweise deutlich ab, das ist systembedingt und wäre nur durch ein wesentlich größeres Phoneminventar kompensierbar. Die Synthese eines durchschnittlichen Satzes von ca. 10 Wörtern benötigt auf aktuellen Computersystemen weniger als eine Sekunde Rechenzeit. Somit ist die Sprachausgabe schnell genug für einen natürlichen Dialog.

Die tatsächliche Realisation der Sprachausgabe findet innerhalb der Care-O-bot 3-Architektur mittels eines Client/Server-Prinzips statt. Dazu wird Festival [1] durch eine (in C++ geschriebene) Serveranwendung gestartet. Der Client, der via Sockets mit dem Server kommuniziert, ist in Python implementiert worden. Der Client kann Text, aber auch Steuerbefehle an den Server senden und ist somit flexibel einsetzbar.

3 Satzgenerierung

Da die gesprochenen Sätze des Roboters situationsabhängig sind, wurde ein Modul zur dynamischen Satzgenerierung entwickelt. Die in Python geschriebene Klasse lädt zuerst ein separates Lexikon, um dann entweder bei der Generierung von Sätzen zu helfen oder automatisch zufällige Sätze zu erzeugen.

Das Programm gibt immer einen String zurück, im Erfolgsfall enthält der String die Äußerung (den Satz), im Fehlerfall ist der String leer. Eine genauere Beschreibung des Fehlers kann dann mit der Methode „getError()“ ermittelt werden, die einen String zurückgibt.

3.1 „sentence()“-Methode

Diese Methode erzeugt aus einer Satzvorlage und übergebenen Variablen einen korrekt flektierten Satz. Beim Aufruf muss als erster Parameter ein String (die Satzvorlage) übergeben werden, die weiteren (optionalen) Parameter enthalten die Variablen. Diese enthalten die Grundformen der Wörter.

Anhand von Beispielen wird die Funktionsweise deutlicher (Ein- und Ausgabe in der Python-Kommandozeile):

```
sentence("Ich _pred011 in _obj01sa!",_pred011="fahren",
        _obj01sa="Zimmer gelb")

>>> 'Ich fahre in das gelbe Zimmer!'
```

```
sentence("_subj01p _pred014 auf _obj01ud.",_subj01p="Tasse",
        _pred014="stehen",_obj01ud="Tisch schwer")

>>> 'Die Tassen stehen auf einem schweren Tisch.'
```

```
sentence("Auf _obj01sd _pred014 _subj01o.",_subj01o="Flasche",
        _pred014="sein",_obj01sd="Regal")

>>> 'Auf dem Regal sind noch Flaschen.'
```

Um die Strings zu erzeugen wird folgendermaßen vorgegangen:

Zuerst wird der erste Parameter (die Satzvorlage) in einzelne Worte aufgesplittet und es werden etwaige Sonderzeichen (Kommas, Punkte, ...) entfernt. Die Variablennamen werden geparkt (s. u.) und anschließend die jeweiligen Werte der Variablen mit Hilfe des am Anfang geladenen Lexikons flektiert und erweitert (z. B. Artikel hinzufügen). Dann werden die Variablennamen durch die soeben erzeugten Strings ersetzt und der fertige Satz zurückgegeben. Sollte ein Fehler aufgetreten sein, so ist ein leerer String der Rückgabewert.

3.2 Syntax

Die Variablen beginnen immer mit einem Unterstrich („_“) und müssen eindeutig sein. Die Reihenfolge der Variablen in der Satzvorlage und in der Parameterliste kann beliebig sein. Es gibt drei Variablentypen: „pred“ (Verben), „subj“ (Subjekte) und „obj“ (Objekte). Zur besseren Unterscheidung wurden die Nomen in Subjekte und Objekte aufgeteilt, die Adjektive sind nicht separat berücksichtigt

worden, sondern werden dem Objekt angefügt. Die Syntax der Variablen ist wie folgt:

Auf den bereits erwähnten Unterstrich („-“) folgt der Variablentyp (z. B. „pred“). Danach wird eine zweistellige ID eingefügt, die nur der eindeutigen Identifikation der Variablen dient. Anschließend folgt je nach Variablentyp (siehe folgende Tabelle) Numerus und/oder Kasus oder Person. Diese Schreibweise ist sehr kompakt und einfach zu parsen.

Im obigen Beispiel ist die erste Variable vom Typ „pred“ mit der ID „01“, das Verb ist 1. Person singular, also „1“. Die zweite Variable ist ein Objekt (mit der ID „01“), welches im Singular („s“) verwendet wird und im Akkusativ („a“) steht. Eine tabellarische Übersicht aller möglichen Kombinationen:

Art	Typ (+ID)	Person	Numerus	Kasus
Verb	_pred(ID)	1, 2, 3, 4	-	-
Nomen	_subj(ID)	-	s, p, o, u	-
Nomen	_obj(ID)	-	s, p, o, u	a, d

Legende: Person: **1, 2, 3** 1., 2. und 3. Person singular
4 Plural
Numerus: **s** Singular
p Plural
o ohne Artikel
u unbestimmter Artikel
Kasus: **a** Akkusativ
d Dativ

Aus der Tabelle erkennt man, dass die Syntax nicht ausreicht, um alle grammatikalischen Möglichkeiten auszuschöpfen, was auch nicht der Anspruch war. Um ein möglichst einfaches System zu bekommen, wurde zum einen vereinfacht (z.B. Reduktion auf vier Personentypen), zum anderen wurde manches nicht für diese Anwendung benötigt (z.B. Genitiv) oder lässt durch andere Formulierungen ersetzen, die das Gleiche aussagen.

3.3 „rand_sentence()“-Methode

Mit Hilfe der obigen Methode lässt sich auch eine zufällige Generierung von grammatikalisch korrekten Sätzen implementieren. Die implementierte Methode „rand_sentence()“ wird ohne Parameter aufgerufen und erzeugt aus intern abgespeicherten Satzvorlagen („templates“) und den Wörtern des Lexikons einen Zufallssatz. Dazu wird zunächst eine Satzvorlage (ähnlich wie bei den obigen Beispielen) per Zufallszahl ausgewählt. Dann werden je nach Anzahl der Variablen weitere Zufallszahlen erzeugt, die als Arrayindex bei dem geladenen Lexikon dienen und je nach Variable somit ein Verb, Nomen oder Adjektiv herausuchen. Mit diesen Informationen wird dann die Methode „sentence()“ aufgerufen und der Satz(string) zurückgegeben. Diese Methode wird hauptsächlich für Tests und Demonstrationen verwendet.

3.4 Ausblick

Da bei der Robotersteuerung sehr allgemein Zustände (stehen, fahren) und Objekte (Schränk, Glas) verwendet werden, ist dieses Modul eine zentrale Schnittstelle, um eine abstrakte Repräsentation in einen natürlichsprachlichen Satz umzuwandeln. Das Lexikon enthält den gesamten Korpus des Serviceroboters Care-O-bot 3.

Bisher wurde das System nur in Simulationen getestet. Der Roboter, für den das Sprachsystem entworfen wurde, befindet sich gerade im Aufbau. Bisherige Erfahrungen haben aber gezeigt, dass eine Sprachausgabe an Servicerobotersystemen die Akzeptanz deutlich steigert. Für die Spracheingabe wird ein kommerzielles Produkt eingesetzt. Um den Einfluss von Umgebungsgeräuschen etc. zu minimieren, wird ein Headset verwendet.

Literaturverzeichnis

1. The Festival Speech Synthesis System (University of Edinburgh, CSTR), <http://www.cstr.ed.ac.uk/projects/festival/> (Retr. Jun 2006)
2. Bernd Möbius. Sprachsynthesysteme. In: Kai-Uwe Carstensen et al. (editors), *Computerlinguistik und Sprachtechnologie: Eine Einführung*, Spektrum, (Heidelberg, Germany), 2001.
3. Alan W. Black and Paul Taylor. Automatically clustering similar units for unit selection in speech synthesis. In: *Proceedings of the European Conference on Speech Communication and Technology* (Rhodos, Greece), volume 2, pages 601–604, 1997.
4. W. J. Wang et al. Tree-based unit selection for english speech synthesis. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)-93*, (Minneapolis, MN, USA), volume 2, pages 191–194, 1993.
5. Andrew J. Hunt and Alan W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)-96*, (Munich, Germany) volume 1, pages 373–376, 1996.