Hauptseminar Dialogsysteme und Voice-XML SS 2011

Christoph Draxler, Klaus U. Schulz

Sitzung 3: Dialog-Management I

Vorlage: McTear Kap. 5

Inhalte und Ziele der Sitzung

Was muss man bei der Steuerungskomponente von Dialogsystemen beachten, welche Alternativen gibt es?

- Dialoginitiative - wer steuert den Dialogverlauf, Mensch, Maschine, beide?
- Drei Arten der Kontrolle des Dialogablaufs (2 näher)
- Einsatz externer Wissenquellen
- Interne Wissenquellen für Dialogmanagement
- Wichtige Punkte bei Entwicklung

Dialoginitiative

Wie ist geregelt, wer wann was spricht?

Drei Arten der Verteilung der Initiative

- System-initiiert
- Benutzer-initiiert
- Gemischte Initiative

Dialoginitiative

System-initiierte Dialogsysteme

Häufigste Form in der Praxis.
Vor allem bei Auskunftssystemen, Transaktionssystemen

System fragt nach der gewünschten Auskunftsart oder Transaktion, nach den hierfür benötigten Daten des Benutzers, lässt sich ggfs. bestätigen, was es erkannt hat, konsultiert Wissensquellen oder DB, erzeugt Antwort.

Vorteile: leicht zu modellieren, System weiß, „wo Dialog steht“, welche Spracheingabe zu erwarten ist. Einfache Spracherkennung.

Nachteile: Rigide, Benutzer hat keine Einflussmöglichkeit, unnatürliche Dialoge, nicht für komplexe Dialoge mit nicht vorhersagbaren Anteilen einsetzbar.

Dialoginitiative

Benutzer-initiierte Dialogsysteme

Z.B. Bei Mehrfachanfragen an DBs
System gibt nur Antworten oder Fehlermeldungen („Verhörbild“).

Vorteil: Benutzer hat aktive Rolle

Nachteile/Probleme:

- Benutzer muss einschätzen können, was System kann.
- Schwierigere Spracherkennung: "alles kann kommen".
- Falls Funktion über die Beantwortung isolierter Einzelanfragen hinausgeht, sehr komplex und schwierig.

Dialoginitiative

Gemischte Initiative

Initiative wechselt zwischen System und Benutzer.
Typisch: Initiative zuerst beim System. Benutzer kann aber komplexere Eingaben machen, als nur die Frage zu beantworten.

S: Wohin wollen Sie fahren?
B: Ich möchte am So nach Augsburg fahren

Vorteile: natürlicher und freier als einseitige Initiative.
Nachteile: schwierigere Spracherkennung, "Gedächtnis" zum Dialogverlauf notwendig.

Dialogkontrolle

Oft drei Hauptkategorien für Kontrolle

- Finite-State-basiert
- Frame-basiert
- Agenten-basiert (nächste Stunde näher, zusammen mit probabilistischer Kontrolle)

Finite-State basierte Systeme

- In den allermeisten kommerziellen Dialogsystemen verwendet.
- Benutzer wird mittels Dialog durch eine Folge vordefinierter Zustände eines Graphen navigiert.
- In jedem Zustand produziert das System einen „Prompt“, d.h. eine für den Benutzer verständliche Äußerung.
- Dieser bestimmt meistens eine Auswahl möglicher nächster Benutzereingaben.
- System erkennt die Benutzereingabe und geht auf dieser Grundlage in einen eindeutig bestimmten Nachfolgezustand, wobei ggfs. eine Aktion durchgeführt wird.
- Initiative nur beim System.

Finite-State basierte Systeme

Beispiel

(Es wird jede Benutzereingabe verifiziert).

System:	What is your destination?
User:	London.
System:	Was that London?
User:	Yes.
System:	What day do you want to travel?
User:	Friday.
System:	Was that Sunday?
User:	No.
System:	What day do you want to travel?

Finite-State basierte Systeme

Beispiel

System: Willkommen bei unserem Informationsdienst!
Welche Art von Information wünschen Sie: Börsenkurse, Sportnachrichten, oder Deutsche Wetterauskunft?
Im ersten Fall sagen Sie bitte "Börsenkurse", im zweiten Fall sagen Sie "Sport", im dritten Fall "Wetter".

Benutzer: Wetter.

System: Für welchen Bereich wünschen Sie eine Wettervorhersage: für Deutschland, Bayern, München, oder für einen anderen Bereich?

.....

Finite-State basierte Systeme

Typisch für finite-state-basierte Systeme

Benutzereingabe auf "kleine" Menge einzelner Wörter oder kurze Phrasen reduziert, die Antworten auf die Prompts darstellen.
Sprachumfang damit von vorneherein kontrolliert.

Vorteile: einfach, hohe Sicherheit bei Spracherkennung.
Nachteile: Behebung von Missverständnissen schwer,

Benutzer kann keine Initiative im Dialog übernehmen.

Sind diese Systeme wirklich durch endlichen Automaten darstellbar?
Nicht im strengen Sinn: Endlicher Automat hat keine Ausgabe! Endlicher Automat hat endliches Eingabealphabet, finite-state basierte Dialogsysteme können u.U. unendlich viele mögliche Eingaben haben (Zahlen).

Finite-State basierte Systeme

Beispiel zustandsbasierter Dialogkontrolle: Nuance Automatic Banking System

Demo System, Englisch, Banktransformationen per Telefon.
Sprachlich „relativ uneingeschränkte“ Eingaben des Benutzers.
Finite-State-Kontrolle kombiniert mit Slotfilling Techniken.

Spezielle Features

- Geldbeträge werden erkannt
- Relative Zeitangaben („on monday“) werden in absolute („April 5th“)
- umgewandelt.
- Synonyme werden erkannt (Midland Bank = HSBC)

Finite-State basierte Systeme

Flexibilität des Ablaufs hängt teils von Aktion ab.

Rechnungsüberweisung: feste Abfolge dreier Schritte:

Empfänger, Betrag, Tag.

Umbuchung auf anderes Konto: flexibler, multiples Slotfillen.

Probleme, wenn Benutzer multiples Slotfillen voraussetzt, dies aber im Dialogzustand nicht möglich ist.

System: Would you like to change the company, amount, or date?
User: The company and the date.
System: Please say company, amount, or date.

Finite-State basierte Systeme

Prinzipielles Problem mit Finite-State-Mechanismus, wenn man

-viel Slots hat,
-multiples Slotfillen ermöglichen will,

Zahl der Zustände und Transitionen wird sehr groß.
Verifikation wird immer schwieriger.

Frame-basierte Systeme

An den Benutzer werden Fragen gestellt, die dazu dienen, Einträge (Slots) eines Frames, eines Formulars, auszufüllen.

Wesentlicher Unterschied zu "klassischen" finite-state-basierten Systemen:

- Dialogfluss ist nicht exakt vorherbestimmt.
- Benutzer kann mehrere Angaben gleichzeitig machen.
- System hält Buch über geklärte und noch offene Angaben.

Frame-basierte Systeme

Beispiel

System:	What is your destination?
User:	London.
System:	What day do you want to travel?
User:	Friday.
System:	At which time do you want to depart?
User:	Around 10 in the morning.

Alternativer Dialogverlauf:

System:	What is your destination?
User:	London on Friday around 10 in the morning.
System:	I have the following connections...

Fragen und Äußerungen des Systems richten sich nach der Menge der ausgefüllten bzw. nicht-ausgefüllten Formulareinträge.
Es bleibt bei einer endlichen Liste von Fragen.

Frame-basierte Systeme

Vorteil:

Flexiblere natürliche Dialoge, gemischte Initiative, aber auf „Formularprobleme“ eingeschränkt. Steuerung der Initiative richtet sich nur nach letzter Äußerung des Benutzers.

Nachteile:

- Die Analyse der Benutzereingabe ist schwieriger, da nicht klar ist, welche Informationen auf einmal gegeben werden und in welcher Reihenfolge.
- Bei numerischen Angaben teils Disambiguierung notwendig. (Telefonnummer oder Kontonummer?)
- Erkennen von Fehlern bei der Erkennung der Benutzereingabe ist schwieriger, auch die Korrektur bzw. der hierzu passende Dialog.
- Spracherkennung anspruchsvoller. Grammatiken notwendig.

Frame-basierte Systeme

Beispiel Frame-basierte Dialog-Kontrolle
Philips Automatisches Zugverbindungs-Auskunftsystem

Offen zugängliches System, Zugverbindungen zwischen 1000 Städten in Deutschland per Telefon: 0241 604020
Deutsche Sprache.
Sprachlich uneingeschränkte Eingaben des Benutzers, fließend.
Vertreter einer Reihe ähnlicher Auskunft-Systeme (Wetter, Börse, ...)

System leitet den Benutzer, damit er alle notwendigen Angaben macht: Startort, Zielort, Abfahrtstag, Abfahrtszeit bzw. Ankunftszeit, welches von beiden?, vormittags oder nachmittags?,...

Frame-basierte Systeme

Beispiel Frame-basierte Dialog-Kontrolle
Philips Automatisches Zugverbindungs-Auskunftsystem

Man könnte System mit Finite-State-Kontrolle ausstatten.
Anderer Extremfall: Natürlichsprachliche Datenbankabfrage, wo der Benutzer alle Angaben in einem Satz machen muss. (Dazu müsste er aber alle notwendigen bzw. möglichen Angaben kennen - unrealistisch!)
System versucht möglichst flexible Zwischenlösung.

System verwendet „Status-Graph“ um bereits gemachte und erkannte Angaben festzuhalten. In Abhängigkeit vom Status bestimmte Reihenfolge der weiteren Fragen.

Frame-basierte Systeme

Beispiel Frame-basierte Dialog-Kontrolle
Philips Automatisches Zugverbindungs-Auskunftsystem

System verwendet verschiedene Strategien zur Verifikation.

- Implizit als „Echo“, gleich mit neuer Frage.
- Explizit als Nachfrage (hält Dialog auf, ist aber manchmal sicherer)

Robustheit: „spät am abend“ wird interpretiert als „zwischen 21 und 23 Uhr“.

Unterschiedliche Ausdrücke für Datumsangaben und Zeiten.
„in diesem Monat“, „drei Tage vor Weihnachten“.

Paralleles Schweizer System wurde evaluiert, über 80% der Benutzer: exzellent.

Agenten-basierte Systeme

Aus der AI-Ecke.

Modellieren eine komplexe Kommunikation zwischen System, Benutzer (und oft auch einer Anwendung), oft um Problem oder Aufgabe zu lösen.

Mehr: Nächste Sitzung

Grounding

Verifikation

System muss Gewissheit haben, die Benutzereingaben korrekt verstanden zu haben. Betrifft alle drei Arten der Kontrolle.
Fast immer wird daher (direkt oder indirekt) nachgefragt, ob Eingabe richtig erkannt wurde.

Zentrales Problem: WANN nachfragen?

- -Nach jeder erkannten Eingabe?
  - Stupide, ermüdender Dialogverlauf
- -Am Ende?
  - Fehleranfällig. Gefahr von größeren Missverständnissen.
  - WAS wurde falsch erkannt? Eine, mehrere, Angaben?

Grounding

Implizite Verifikation

Teils wird die Verifikation „unauffällig“ als Statement ausgedrückt und mit nächster Frage verbunden:

„Sie wollen also nach Hamburg fliegen. An welchem Tag?“

Sollte tatsächlich ein Fehler vorliegen, ist aber die Benutzerreaktion schwer einzugrenzen.

„Nein“
„Nein, nicht nach Hamburg“
„Nein, das haben Sie falsch verstanden“
„Ja, im Prinzip schon, aber ...“
„Nach HOMBURG!!“

Grounding

Komplexere Grounding-Probleme

Es kann Unterschiede geben zwischen Info, die Benutzer haben möchte, und Info, auf die System zugreifen kann.

Es kann in anderen Punkten falsches Bild des Systems und seiner Möglichkeiten beim Benutzer geben, das korrigiert werden muss.

Vgl. Ausführungen zu Grounding in Vorsitzung.

Zugriff auf externe Wissenbanken

Dialogsysteme konsultieren oft Wissens- oder Datenbanken

Probleme

- Vokabular in der Anfrage muss genau auf das Vokabular in der Datenbank matchen.
- Matches in der Datenbank und mögliche Antworten können ambig sein („Frankfurt“).
- Anfragen können aus anderem Grund unterspezifiziert sein.
- Wie große Antwortmengen dem Benutzer präsentieren?

Wissensquellen für Dialogmanagement

Manchmal als Teil des „Dialogmodels“ aufgefasst.

- Dialog-Geschichte (Anaphernresolution, Grounding,…).
- Aufgaben-Liste (Verbleibende Aufgaben und noch notwendige Angaben)
- Modellierung von Weltwissen (Bsp. Interpretation von Zeit- und Ortsangaben, „Common-Sense Reasoning“)
- Domänen-Modell
  Architektur eines Gebäudes..
  Thematische Bereiche bei Berufsfeldern.
  Kenntnis aller Flugverbindungen...

Wichtige Punkte bei der Entwicklung von Dialogsystemen

- Zusammenstellen von Ressourcen
  Spracherkenner, Grammatikumgebung, Spracherzeugung

- Wahl der Steuerung der Dialog-Initiative
  System, Benutzer, gemischt? Adäquatheit für gegebene Anwendung. Geeignete Ressourcen (Spracherkennung, Sprachverstehen, Spracherzeugung) vorhanden?

- Wahl der Dialog-Kontrollstrategie
  Finite-State, frame-basiert, agentenbasiert? Durch Dialoginitiative und Ressourcen eingeschränkt.

- Design der System-Prompts
  Ziele: klarer natürlicher Dialog-Ablauf, Menge der möglichen Äußerungen einzuschränken, um Spracherkennung leichter zu machen

Wichtige Punkte bei der Entwicklung von Dialogsystemen

- Wahl der Verifikations-Strategie
  Explizit versus implizit, direkt versus kombiniert
- Fehlererkennung und Fehlermanagement
  Welche Fehler können auftreten, wie feststellen, wie behandeln?
- Design und Implementierung für Zugriff auf externe Wissensquellen
  Vokabularüberprüfung, Synonymlexika etc.

Wichtige Punkte bei der Entwicklung von Dialogsystemen

Einfaches versus komplexes anspruchsvolles System

In der Praxis entscheiden zumindest zwei Kriterien über die Akzeptanz eines Dialogsystems

1. Wie natürlich und frei sind die Dialoge?
2. Ist für den Benutzer klar nachvollziehbar, was das System kann und was nicht, wie es sich verhält?

Raffinierte und natürliche Dialoge lösen Anspruchsdenken des Benutzers aus. Problem, wenn an anderer Stelle oder in bestimmten Situationen Erwartungen nicht erfüllbar sind und letztlich dann sehr oft unklar bleibt, was vom System erwartet werden kann.