GoDiS and TrindiKit

MITRE workshop 27/10-03

Staffan Larsson
Göteborg University
Sweden
TrindiKit and GoDiS

• TrindiKit
  – Toolkit for building and experimenting with dialogue move engines (dialogue managers)
  – Information state update approach
    • Information states
    • Update rules
    • Dialogue moves

• GoDiS
  – A dialogue system implemented with TrindiKit
  – Issue-based dialogue management
  – Versions for Inquiry (IOD) and Action-oriented dialogue (AOD) genres
Dialogue task and source of back-end information

- **Task**
  - Medical diagnosis
  - Information about diseases

- **Source of back-end information**
  - Java backend application
  - Connected to GoDiS using OAA
  - Not just a database
    - Also provides which questions to ask and which tests to perform
    - Dynamic dialogue planning

- **Type of dialogue**
  - Inquiry-oriented
  - System-driven dialogue
  - Single task (but more can be added)
Modules we needed to change for this exercise

• Resources
  – Domain knowledge resource
    • Dialogue plans
    • Ontology
  – Device interface
    • Connection to Java database via OAA
  – Lexicon
    • Includes synonyms, extracted from backend

• Modules
  – Some minor bug-fixes and improvements
  – These are not domain-specific, and will thus be reused in other domains
Domain knowledge

• Dialogue plans for dealing with diagnosis
  – ?x.disease(x): “which disease is diagnosed?”
  – ?confirmed_by_interview: “Is the diagnosis confirmed by additional information?”
  – ?confirmed_by_tests: “Is the diagnosis confirmed by medical tests?”

• These issues are all assumed to be open when dialogue starts

• Additional plans
  – ?x.info(x): “What information is there about a given disease?”
  – ?x.treatment(x): “What treatment is there for a given disease?”
Alternate solutions

• User-driven dialogue
  – Not load issues when resetting; user has to raise all issues
  – Usr can ask sys to
    • Provide a diagnosis
    • Confirm whether usr has give disease

• Decision trees as dialogue plans
  – Move backend knowledge into dialogue plans
  – Information conversion could be done automatically

• Separate genre: expert system dialogue
  – Add special purpose update rules
  – Dynamic dialogue planning by expert
What was added to GoDiS?

• 2 update rules
  – exec_assume
    • If assume(P) on top of plan, add P to /SHARED/COM
  – exec_assume_issue
    • If assume_issue(Q) on top of plan, push Q on SHARED/ISSUES

• Possible useful additions
  – Subplans
  – Precedence constraints on tasks
Highlights

• Reuse of GoDiS -> you get for free:
  – Feedback, grounding, sequencing
    • Though this is more useful with speech
  – Accommodation and clarification
    • incl. simple plan recognition (dependent issues)
  – Multiple simultaneous issues & information sharing between plans

• High level of abstraction in dialogue plans

• Modularity -> rapid prototyping

• Separation of lexicon from domain knowledge
  – Additional languages can be easily added, and language can be changed online
Lowlights

• GoDiS has not been used for this type of application, so not completely straightforward
  – Backend, not dialogue plans, determines which question should be asked next
  – Solution: each plan is loaded several times, until goal issue has been resolved
• Generation and interpretation a bit tacky
• Had to use AOD version to connect to device
  – Need to make an IOD version for devices
• Accommodation has to be helped with “bind”
  – Suggests different datastructure for plans
EXTRA SLIDES
Accommodation and clarification

S> Are you feeling nauseous?
U> yes and i have a headache [score 0.0-1.0]:0.8
S> Okay. headache.
U> yes

S> Hello
U> malaria
S> malaria. I dont quite understand. Do you want to confirm by interview , to confirm by taking tests , to get general information or to ask about treatment?
Tested with real users?

• So far, not much
  – A little bit done by students
• Starting soon (before Xmas)
  – Corpus collection
  – Application: VCR programming
Multiple simultaneous issues, information sharing

S> Hello How are you feeling today?
U> I have a headache
S> Okay. Are you feeling nauseous?
U> Um… can I have some information about a disease?
S> Okay. Let's see. What disease do you want to know about??
U> I also want to know about treatment
S> Okay. Let's see. What disease do you want to know about??
U> malaria
S> Malaria can be cured with prescription drugs...
S> Returning to the issue of information. Malaria is a serious, sometimes fatal, disease caused by a parasite...
S> Returning to the issue of your diagnosis. Are you feeling nauseous?
GoDiS-AOD information state

PRIVATE:
- AGENDA : stack( Action )
- PLAN : stackset( Action )
- BEL : set( Prop )
- TMP : (same type as SHARED)

SHARED:
- COM : set( Prop )
- ACTIONS : stackset( Action )
- QUD : stackset( Question )
- ISSUES: stackset( Question )
- LU:...
TrindiKit:
How to build a system
How to use TrindiKit

• We start from TrindiKit
  – Implements the information state approach
  – Takes care of low-level programming: dataflow, datastructures etc.
How to build a basic system

- Formulate a basic dialogue theory
  - Information state
  - Dialogue moves
  - Update rules
- Add appropriate modules (speech recognition etc)
How to build a genre-specific system

- Add genre-dependent IS components, moves and rules

genre-specific theory additions \(\rightarrow\) genre-specific system

basic dialogue theory \(\rightarrow\) basic system

information state approach \(\rightarrow\) TrindiKit
How to build an application

- Add application-specific resources

- domain & language resources
- genre-specific theory additions
- basic dialogue theory
- information state approach

application
genre-specific system
basic system
TrindiKit
GoDiS-I

GoDiS-A

GoDiS

TrindiKit

IBDM

Travel Agency

application-specific

Auto-route

Xerox manual

home device manager

VCR manager

genre-specific

IS approach

Goteborg University Dialogue Systems Lab