But, is human-like dialogue ability really needed?

- Turing Test: Instantiation of the Turing Test
  - Became known as the Turing Test
  - If we can build a computer system that can carry on a conversation and we cannot tell it is a computer, we have achieved artificial intelligence.
  - Award given to most human-like computer.

Turing and Loebner Contests

Overview

⇒ Introduction
  - FSMs
  - Slightly beyond FSMs

⇒ Terminal Substitution
  - What is Semantics?
  - Interpretation
  - Example Grammars
  - Regular Grammars
  - Need for Phrases
  - Introduction

⇒ Interpretation

⇒ What is Semantics?
Emphasis is on the technologies and how to put them together.

- Understanding approaches that can be used to allow us to build more
- Understanding their advantages and disadvantages
- Understanding how they work
- Start with simple systems (and components)
  
In the course,

- Start with simple systems (and components)
  
But might be difficult for some to user
- The less we give the user, the simpler the system is
- How much flexibility should we give the user?

Practical Dialogues

Practical Dialogue Hypothesis (Allen et al., 2001, AI Magazine)
The conversational competence required for practical dialogues,
although still complex, is significantly simpler to achieve than general
human conversational competence.

Don’t need to build a system
- that can fully understand all the nuances of what you say
- that can converse about poetry
- need to build a system
- successfully is user accomplished the goal in a reasonable amount of time
- conversations about some task

Practical Dialogues
Components of a Spoken Dialogue Systems

- Speech recognition
- Convert audio signal into a sequence of words
- Understanding
- Map word sequence into a language that the system understands
- Reasoning
- Decide what to display in response to user's speech
- General response
- Map system's response into a sequence of words
- Generate response
- Convert sequence of words into audio signal

But none of these problems are solved.

Varying Complexity of Spoken Language Systems

- How complex the system's actions are
- How complex user utterances are
- How indirect the actions are depends on the user's response.

From DeMori 2003 HTL/NAACL dialogue workshop.

Speech Recognition

Understanding

Reasoning

Action
Course Info

- Final Exam 35%
- Presentation 15%
- Assignments 50%

Bulletin Board: http://www.cslu.ogi.edu/forum

heemanp@ohsu.edu

http://www.csee.ogi.edu/class/cse550

Syllabus

Finite-State Dialogue Management
- Separating ASR from Parsing and Semantics

Form-based Dialogue Management
- Rule-based Dialogue Management
- Feature-State Dialogue Management
- Separating ASR from Parsing and Semantics

Machine Learning of Dialogue Policies
- Simulated Users, Objective Function, Reinforcement Learning

Special Topics
- Grounding, Turn-taking, Adjacency Pairs, Initiative, Discourse
- Speech Act, Information State Update
- Speech, Affects, Dialogue Policies

Structure, Application
Overview

- Terminal Substitution
- What is Semantics?
- Interpretation
- Example Grammars
- Regular Grammars
- Need for Phrases
- Slightly Beyond FSMs
  - FSMs
  - Introduction

Academic Integrity

- It is acceptable to discuss the general concepts and principles behind the assignment and solution so as to be able to explain it adequately to the instructor.
- Each student is expected to develop, write up, and hand in an individual solution and, in doing so, develop a sufficient understanding of the problem and solution so as to be able to explain it adequately to the instructor.
- It is not proper to arrive at collective solutions.
- It is acceptable to discuss the general concepts and principles behind the assignment with other students.
Simple Dialogue System

- Canned dialogues
  - Where system follows a set script
  - User just fills in some blanks here and there
  + Answers yes or no, or from a very limited set of responses
  - List of facts fills in some blanks here and there
  - Where system follows a set script

How to specify the script (or dialogue behavior)?

- Use a flowchart?

Minimize impact of

- Interpretation of what is said
- Limitation of speech recognition technology

Dialogue Flow

- To accomplish some goal, will take a number of speaking turns
  - User says something
  - System says something
  - User says something
  - System says something
  - ...
User responses

- User must give information in order that system wants
- User must give information one piece at a time
- User's actions are very restricted
- Each response has simple mapping to meaning
- Recognizer constrained to allowed responses
- Responses usually just a single word
- Each response has a simple mapping to meaning
- Transition: allowed responses
- Master-slave dialogue, with system as master and user as slave
- Dialogue driven by the system
- Context-dependent speech recognition
- Typically only allow a small number of allowed user responses

FSM for Dialogue

- Transitions can have a user input
- States can have a system output
- Set of transitions between states
- Set of states
- What is a finite state model?
- Specifies user actions
- Specifies system actions

What happens in a dialogue specified by a finite state model?
Advantages of FSM for Dialogue Control

• Advantages of FSM:
  
  - Speech recognizer uses dialogue state
  
  - Just has to distinguish between a few alternatives
  
  - Easy to understand, how-choos, of what can happen
  
  - Speech synthesis can use "canned phrases"
  
  - Speech recognizer uses dialogue state

• Disadvantages of FSM:

  - Forces user to segment what they want to do into the pieces of the
    dialogue
  
  - Where system can ask questions, and user provide short answers

Best Practices

• Keep number of possible user responses as small as possible

• System prompts should elicit as few variations as possible

- System prompts should elicit as few variations as possible

  - If special commands, such as cancel, should be in all states, and user

  - Should be made aware of this

  - If special commands, such as cancel, should be in all states, and user

  - System prompts should elicit as few variations as possible

• Keep number of possible user responses as small as possible

  - User could say "man", "male", "masculine", ...

  - Don't ask "what is your gender?"

  - Include all genders, or just two choices:

• Keep things consistent, transparent, and easy-to-use

  - System prompts should elicit as few variations as possible

  - Ask "are you male or female?"

  - User could say "man", "male", "masculine", ...

  - Don't ask "what is your gender?"

  - Include all genders, or just two choices

  - Keep number of possible user responses as small as possible
Structured Dialogues with Variables

• To make structured dialogues more compact, allow variables
  - Recognized responses saved in variables
  - System responses can use variables (template-based speech generation)
  - In fact, can allow arbitrary code

  That is a $s1(recog)\ s2(recog)$

  pizza
ev
  ve
  g
  g
  e
  e
  i
  i
  c
  c
  e
  e
  r
  r
  a
  a

  Thank you

  small
  large

  $s1:\ s2(recog)$

  What size pizza do you want?
  What kind?

  States can execute arbitrary code
  Transition can be based on variable settings, not just recognized responses
  - Slightly beyond FSMs

Overview

• Terminal Substitution
• What is Semantics?
• Interpretation
• Example Grammars
• Regular Grammars
• Need for Phrases

  ⇒ Slightly beyond FSMs
  ⇒ FSMs
• Introduction
Might want common routines for street address, city and zip code, or amounts could reuse them in any application for street address, city and zip code, or amounts.

• With RTN's can have node that is itself an RTN. Equivalent to Content Free Grammar.

Recursive Transition Networks

Banking Application

• User can transfer, withdraw
  balance withdraw transfer
  type of transaction?
  account? account? from?
  to? amount? balance is ...
  s3 s7 s10 s5 s4 s6
  password?
  Exit
  userid?
  s1
  s8 s11
  amount?
  s12
  balance in $s9(recog) is ... and $s10(recog) is...

• Difficult to do the amounts.
  For now, only allow amounts of $20, $40, $60, or $80.
  • User can transfer, withdraw money and find out account balance.
Overview

Introduction

FSMs

Slightly beyond FSMs

Need for Phrases

Regular Grammars

Example Grammars

Interpretation

What is Semantics?

Terminal Substitution

Powerful mechanisms (including variables and RTN's)

Question: Is FSM a good formalism for expressing dialogue behavior?

Even though Powerful...
What is Needed?

- Recognize phrases

  - Table lookup would also require 9999 items!
  - Infeasible to list all possibilities and treat them all as individuals words

- Interpret phrases

  - Infeasible to list all possibilities and treat them all as individuals words

  - For numbers between 1 and 9999, there would be 9999 items!
  - Even more, since different ways of saying same number

Recognize phrases

Banking Application

- User can transfer, withdraw balance

  - What type of transaction?
  - From account?
  - To account?
  - Amount?

  - Password?

  - Exit

- Difficult to do amounts

  - Could add extra states

  - How many one's?
  - How many ten's?
  - How many hundreds?
  - How many thousands do you want?

  - Balance:

  - User can transfer, withdraw money and find out account balance.

  - CSE560 Class 01: 2003
Overview

Introduction

FSMs

Slightly beyond FSMs

Need for Phrases

Regular Grammars

Example Grammars

Interpretation

What is Semantics?

Terminal Substitution

Allow User to Over-answer

• Would also be good to allow user 
  balance withdrawl transfer 
  type of transaction?

account? account? from?

to? amount?

balance is ...

password?

Exit

userid?

balance in $s9(recog) is ... and $s10(recog) is...

• When asked transaction type,
  allow user to also say other parameters
  i.e. transfer from checking to savings
  i.e. balance of checking account

• If user specifies extra parameters,
  allow user to also say other parameters

• Would also be good to allow user
  to specify more things at once

⇒ Toward Mixed-initiative dialogues

⇒ Skip to appropriate state

⇒ User can take initiative to add extra

• Information that was not requested

⇒ User can take initiative to add extra
Right-Linear Regular Grammar

- Start state
- Non-terminals (For clarity: tokens that start with $)
- Terminals (For clarity: tokens that start with letter)
- Non-terminals (For clarity: tokens that start with $)

Rules of form:

- Non-terminal = 0 or one non-terminal followed by 0 or more terminals
- Similarly, can define Left-Linear Regular Grammars

- Right Linear Regular Grammar formalism is simple, but awkward.

Speech Recognition of Phrases

- Need way to concisely represent what recognizer can accept
- Should take into account what things make sense, what doesn't make sense and probably was not said
-Recognizer in CSLU toolkit allows regular grammars
- Same expressive power as finite state machines and regular expression
- CSLU toolkit allows regular grammars
- Different forms for writing regular grammars
- All have same power
Delimiters

- Only one rule can be used to define non-terminal
  - Right hand side can include delimiters
    - Right Linear Regular Grammars don't have this restriction

CSLU Toolkit Notation

- Non-terminals start with "$"
- A grammar rule
  - Any sequence of terminals or non-terminals on right hand side
  - Non-terminal on left hand side
  - "$" used to separate left-hand side from right hand-side
  - RAD Tutorial 15
Grammar For Amounts

- Recognize digits one to nine
  \[ \text{OneToNine} = \text{one} | \text{two} | \text{three} | \text{four} | \ldots | \text{nine} \]

- Recognize numbers from ten to nineteen
  \[ \text{Teens} = \text{ten} | \text{eleven} | \ldots | \text{nineteen} \]

- Recognize numbers from twenty to 99
  \[ \text{Tys} = \text{twenty} | \ldots | \text{ninetynine} \]

- Recognize numbers from one to 99
  \[ \text{Tens} = \text{Tys} | \text{Teens} | \text{OneToNine} \]

- Recognize numbers from 100 to 999
  \[ \text{Hundreds} = \text{OneToNine hundred} [ \text{Tens} ] \]

- Recognize numbers from 1000 to 9999
  \[ \text{Thousands} = \text{OneToNine thousand} [ \text{Hundreds} ] \]

- Recognize from 1 to 9999 dollars
  \[ \text{Amount} = ( \text{Thousands} | \text{Hundreds} | \text{Tens} ) \text{ dollars} \]

Ensuring Regularness

- Rules are ordered
- Non-terminals in right-hand side must already be defined
- Rules are ordered

- CSTL formalism can be mapped into a right-linear regular grammar
- But equivalent in power
- Useful if there are efficient algorithm for right-linear regular grammars

- Easier to write rules in this notation than as a right-linear regular grammar
- CSTL formalism can be mapped into a right-linear regular grammar

- Rules are ordered
- Non-terminals in right-hand side must already be defined
- Rules are ordered

- Rules are ordered
Finite State Machine for Numbers

From Automata, FSM is equivalent to regular grammars - Can convert the regular expression for Ty's to an FSM.

Example of FSM machine for Ty's

• So, can use FSM's for dialogue states and word sequences

• Can convert the regular expression for Ty's to an FSM.

• Finite State Machine for Numbers

Full Grammar

• Not only concise, but easy to debug, and maintain

7 Rules to capture 9999 different numbers:

4 dots ( one | two | three | four | five | six | seven | eight | nine | ten | eleven | twelve | thirteen | fourteen | fifteen | sixteen | seventeen | eighteen | nineteen | twenty | thirty | forty | fifty | sixty | seventy | eighty | ninety ) dollars

$OneToNine = one | ... | nine

$Tys = ( twenty | ... | ninety ) $OneToNine

$Tens = $Tys | $Teens | $OneToNine

$Hundreds = $OneToNine hundred $Tens

$Thousands = $OneToNine thousand $Hundreds

$Amount = ( $Thousands | $Hundreds | $Tens ) dollars
Home Automation Example

- User must still learn allowed syntax
  - Speech recognition harder now?
  - One node that allows user to ask any question

FSM for dialogue structure could be pretty simple

• Turn up music in dining room
• Turn off music in living room
• Turn on light in kitchen

Commands to system

User must still learn allowed syntax
  - Speech recognition harder now?
  - One node that allows user to ask any question

FSM for dialogue structure could be pretty simple

• Turn up music in dining room
• Turn off music in living room
• Turn on light in kitchen

Overview

- Terminal Substitution
- What is Semantics?
- Interpretation
- Example Grammars
- Regular Grammars
- Need for Phrases
- Slightly beyond FSMs
- FSMs
- Introduction
Aside: How Many Variations are Needed?

- So many variations...
  - Do test runs to find out if coverage is adequate
- Larger search space recognizer is searching over
  - Slower it will become
  - Greater chance it will misrecognize
- Do test runs to find out if performance is good enough
  - Collect a bunch of samples from lots of people
  - Measure speech recognition performance

Best practices for how to design prompts to encourage standard way of saying things:

- Collect a bunch of samples from lots of people
- Measure speech recognition performance
- Do test runs to find out if coverage is adequate
- Collect a bunch of samples from lots of people
- Measure speech recognition performance
- Do test runs to find out if performance is good enough

Banking Example:

- transfer twenty dollars from checking to line of credit
- transfer twenty dollars to savings from checking
- transfer from checking to savings thirty dollars
- transfer forty dollars to my savings from checking
- transfer from checking to savings thirty dollars
- transfer twenty dollars from my savings
- withdraw 20 twenty dollars from my savings
- balance of my savings account
Speech recognizer returns sentences of form "turn on light in kitchen".

Home Automation Example

```plaintext
set room [join $reco 3 end]
set device [index $reco 2]
set state [index $reco 1]
Could extract third and following words third word as device
Could extract third word as device
Could extract second word as state
```

Overview

• Introduction
• FSMs
• Slightly beyond FSMs
• Need for Phrases
• Regular Grammars
• Example Grammars
• What is Semantics?
• Terminal Substitution
• Interpretation
• What is Semantics?
We need a principled way of doing this.

- Speech recognizer returns strings of form: "nine hundred seventy nineteen two thousand three hundred forty five"
- Will be worse if we allowed user to incorporate amounts in sentences like "transfer three hundred twenty five dollars from checking to savings"
- Code to build the amount will be very cumbersome to write

We need a principled way of doing this!

Speech recognizer returns strings of form: "nine hundred seventy nineteen two thousand three hundred forty five"

Amounts in Banking Example

- Did student 555 take CSE550?
- What was the grade of student 555 in CSE550?
- How many students took CSE550 in Summer 2002?
- What was the highest mark in CSE550 in Summer 2002?
- When was CSE550 last taught?

Student Database Example

- Did student 555 take CSE550?
Amounts in Banking Example

- We still need to map words to meaning
  - What does "three thousand two hundred twenty five" mean?
  - For our banking application, it is a monetary amount that we will transfer or withdraw.
  - We still need to map words to meaning.

Overview

- Introduction
- FSMs
- Slightly beyond FSMs
- Regular Grammars
- Example Grammars
- Need for Phrases
- Terminal Substitution
- What is Semantics?
- Interpretation

Interpretation
Semantics

Semantics is the internal representation that the computer converts the human language into, so it can easily reason with it.

- Unlike human language, non-ambiguous
- Expressive enough to capture what the computer needs to reason about
- But collapses differences that are too subtle for the computer

Is there a principled way of building the appropriate semantic representation from the words?

"Can you turn on the light?" versus "Turn on the light."

But collapses differences that are too subtle for the computer.

Home Automation Example

Speech recognizer returns sentences of form "turn on light in kitchen."

Want to map the words into a simpler representation that a computer can process.

Turn(State,Device,Room)

Could use following representation:

- Could pass this to a procedure that executes this:
  - turn (State, Device, Room)

Could refer to the formalism for building the semantic representation as the Semantic Building Formalism.

Or just use: State Device Room
Simple Semantic Processing

For home automation, let's use a representation language of form:

- For home automation, let's use a representation language of form:

  light kitchen ⇒ kitchen
  living room ⇒ livingroom
  on ⇒ 1
  off ⇒ 0
  turn ⇒ ǫ
  in ⇒ ǫ

Terminal Substitution Formalism

- Have speech recognizer output the wanted symbols instead of the actual words that were said.

  Terminal Substitution Formalism

  • in ⇒ in
  • turn ⇒ turn
  • living room ⇒ livingroom
  • kitchen ⇒ kitchen
  • light ⇒ light
  • off ⇒ off
  • on ⇒ on

So, semantic building language formalism needs to map:

on high kitchen

- Slightly beyond FSMs
- Need for phrases
- Regular grammars
- Example grammars
- Interpretation
- What is Semantics?

Overview
Incorporating into Regular Grammars

- Terminal Substitution formalism can be embedded into the grammar rules
  - CSLU Grammar Constructs
    - `%%` When following a word, will skip the word in the recognition results if the first word is recognized
    - `%' When following a word, will substitute the next word into the recognition output

Example

```plaintext
$room = (living%livingroom room%%) | kitchen | ...
$command = turn%% $state $device in%% [ the%% ] $room
```

The recognition output:

| When following a word, will skip the word in the recognition results if the first word is recognized | `%%` |
| When following a word, will substitute the next word into the recognition output | `%` |