NPFL099 - Statistical dialogue systems

Building an SDS

Filip Jurčíček

Institute of Formal and Applied Linguistics
Charles University in Prague
Czech Republic

Home page: http://ufal.mff.cuni.cz/~jurcicek

Version: 26/02/2014

Outline

- Typical behaviour of SDS
    - Confirmation
    - Grounding
    - Back-channeling
    - Timing
- Building a new SDS
- State based dialogue systems
- VoiceXML

Typical behaviour of SDS

- Help
  - User can ask for help at any time
  - Provide info on where they are in the dialogue
  - Give an example what they can say

- Errors
  - Sorry, I did not understand.
  - Sorry, I did not get that quite right.

- Confirmation
  - Did you say cheap?

Confirmation

- There are two types
- Explicit confirmation
  - What type of venue are you looking for?
    - Restaurant
  - You said restaurant, did I get that right?
    - Yes
- Implicit confirmation
  - What type of venue are you looking for?
    - Restaurant
  - Restaurant, in what area do you want to eat?
    - ... user can barge-in ...

Explicit vs. implicit confirmation

- Explicit confirmation
  - Safe but slow

- Implicit confirmation
  - Natural, it is difficult to support interruptions from users

Prompt design

- Constrain your questions:
  - How may I help you?
    - User replies with a long story

- What type of food do you want?
  - System can expect food type

Grounding

- Definition
  - Showing evidence of understanding / confirmation

- Example
  - What type of venue are you looking for?
    - Restaurant
  - **good:** Right. In what area do you want to eat?
  - **better:** Restaurant, right. In what area do you want to eat?

Back-channeling

- Definition
  - Back-channelling is a way of showing a speaker that you are following what they are saying and understand, often through interjections like *I see, yes, OK* and *ehm.*
- Most SDSs does not do it
  - Filler words - uhms, errs,
  - Yeah, right, ok
- Human-machine communication is much more restricted
  - A potential source of inefficiency in conversation.

Timing

- Humans replies before the end of the partners turn
  - Humans successfully predict when a turn ends
    - prosody, meaning
- Humans integrate multiple sources of information, e.g. speech and vision
  - vision can help with voice activity detection (eyes and mouth tracking), e.g. Kinect
  - sound localisation, beam forming when using microphone arrays

Building a new SDS

- Define the domain/task
  - Collect several dialogues in the domain
- Analyse what information is exchanged
  - Address (of a venue, departure or arrival)
  - Time (of departure, ...)
  - Date (if a booking)
  - Location (of a venue)
  - Price (of a meal, a bus ticket, airfare, ...)

Build an ontology

- Definition
  - A set of domain specific information describing the task.
- Ontology includes
  - Concepts used by the SDS
    - e.g. bus stops, times, food types, price range
  - Values for all domain concepts
  - How to say them
- The ontology is used to derive the dialogue state and very often the system actions.

How the SDS will react to user input

- Define the system actions, e.g.
  - questions, offers, confirmations
- What is a typical/best order of questions
- How to request missing pieces of information
- How to confirm uncertain/contradicting information

Build a handcrafted system first

- Simple handcrafted systems are quick to build. Though, may not be optimal.
  - SLU - rule based
  - DM - frame based dialogue state monitoring
  - DM - decision tree policy
  - NLG - template based generation
- Test without ASR or even without SLU and NLG
- Typing is easier then talking
  - you can type text, or
  - type directly dialogue acts

Testing the system

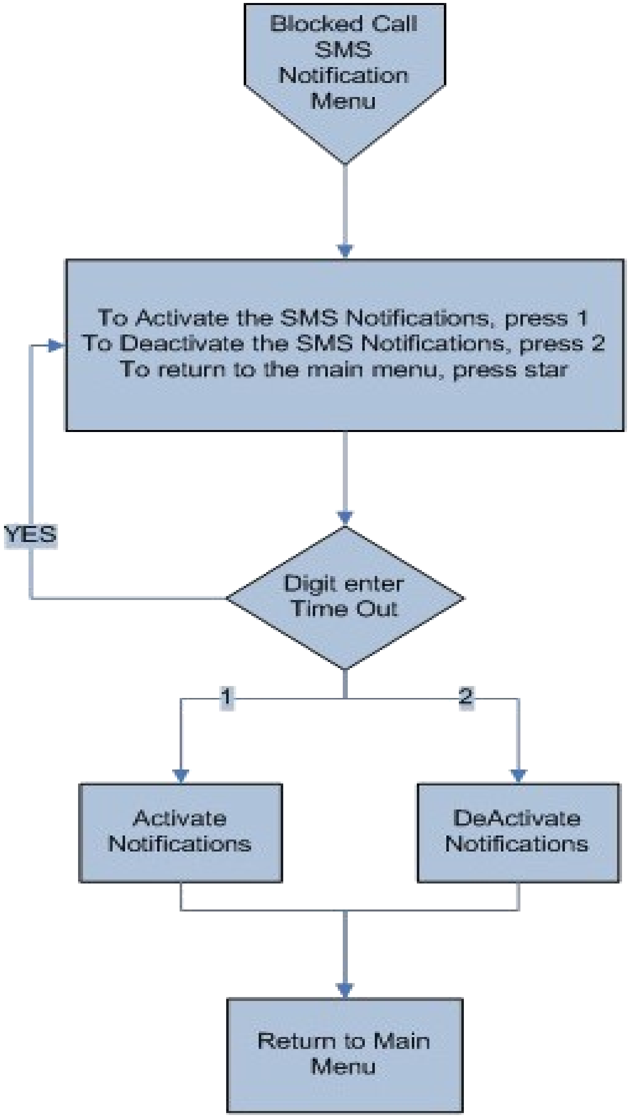
- Test all supported scenarios
- Test quality of ASR - e.g. LM
- Test robustness of SLU
- Ask others to test the dialogue system
- Use Amazon Mechanical Turk

Design of the help

- Be consistent and concise
  - Good examples of what a user can say
  - Provide variations of examples
  - Provide different levels of help
    - At first, be concise
    - The second time, provide detailed help
- Is the provided help really useful?

State based dialogue systems

- A designer typically define a dialogue flow


Blocked Call SMS Notification Menu
To Activate the SMS Notifications, press 1
To Deactivate the SMS Notifications, press 2
To return to the main menu, press star
YES
Digit enter Time Out
1
2
Activate Notifications
DeActivate Notifications
Return to Main Menu

Example: Get an account balance

- Sub-dialogues
  - Get name
  - Get account number
  - Get pin
  - Provide balance
  - Exit

Sub-dialogue: Get name

- Prompt: What is your name?
  - Recognize the name
  - May be correct (in the database)
  - May be unknown (not in database)
  - May not be name (What do I say? Help/Repeat)
  - Should you confirm the recognised name?

Dialogue flow

- Get name
  - Check in database
  - Ask again if not
  - Deal with help
- Get account number
  - Check in database (with name)
- Confirm account number and name
  - For security

Complexity of dialogue flows

- Dialogue flows can get large
  - User can get lost
- Minimize the number of turns
  - Shorter calls result in happier customers
  - Shorter calls result in more calls

First time users

- First time users need a successful call
  - Otherwise they will not call back
- The way you write your prompts is important
  - Different wording elicit different answers
  - Users can get confused

VoiceXML

- The W3C's standard XML format for interactive voice dialogues
  - XML language for SDS (like HTML)
  - Input:
    - Recognised speech
    - Touch-tone aka DTMF
  - Output:
    - Synthesised speech
    - Recorded speech

VoiceXML: ASR

- Using a n-gram language model
- Using grammars (JSGF,SRGF)
  - Define accepted input such as
    - Account numbers
    - Cities
    - Phone numbers
    - Dates

VoiceXML: TTS

- <ssml> xml
- Pick a voice
- Pick a language
- Define pronunciation of unusual words, e.g. names
- Define prosody in prompts

VoiceXML example 1

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form>
   <block>
     <prompt>
      Hello world!
     </prompt>
     <goto next="#say_goodbye"/>
   </block>
  </form>

  <form id="say_goodbye">
    <block>
    <prompt>
      Goodbye!
    </prompt>
    </block>
  </form>
</vxml>
```

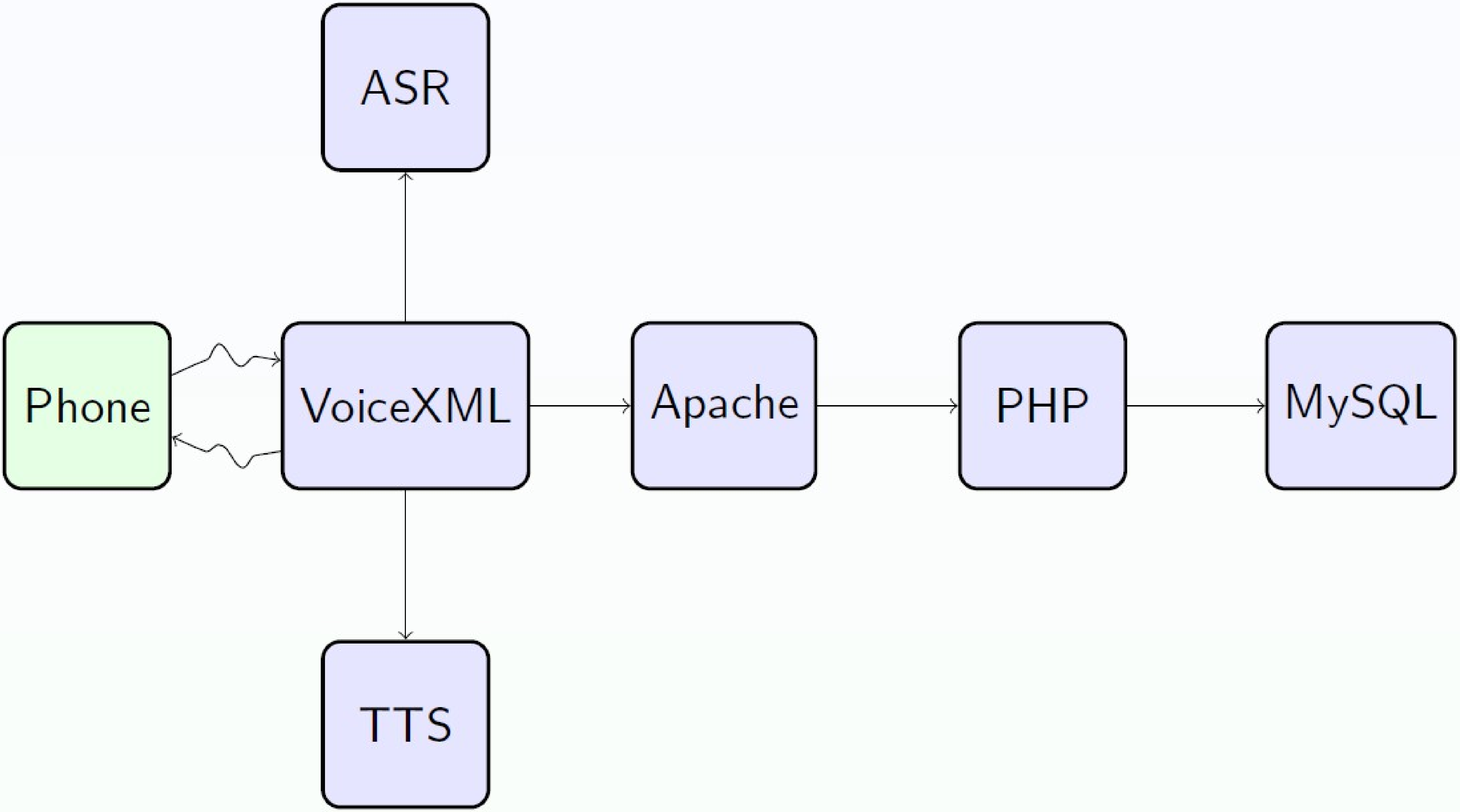
Basic tags

- <form id=``XXX"> like a form in HTML (a group of inputs)
- <field> input from user trough speech or DTMF
- <record> record users input
- <subdialogue> performs sub-dialogue
- <goto next=``continue.vxml"> moves execution to the next VoiceXML file

VoiceXML example 2

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form id="weather_info">
  <block>Welcome to the weather information service.</block>
  <field name="state">
    <prompt>What state?</prompt>
    <grammar src="state.grxml" type="application/srgs+xml"/>
    <catch event="help">
      Please speak the state for which you want the weather.
    </catch>
  </field>
  <field name="city">
    <prompt>What city?</prompt>
    <grammar src="city.grxml" type="application/srgs+xml"/>
    <catch event="help">
      Please speak the city for which you want the weather.
    </catch>
  </field>
  <block>
  <submit next="/servlet/weather" namelist="city state"/>
  </block>
</form>
</vxml>
```

SDS using VoiceXML



VoiceXML browsers

- Use PHP to generate VoiceXML files
  - Use urls (with ?...) to calculate/get data (http://weather.com?zip=”15213")
  - Use urls to get waveforms (http://tts.com?text=”Hello World")

- VoiceXML browsers
  - Commercial
    - Nuance
    - KKY, ZCU v Plzni
- Open source
    - OpenXMI - http://www.speech.cs.cmu.edu/openvxi/
    - JVoiceXML - http://jvoicexml.sourceforge.net/news.php

Summary

- We explained today
  - Typical behaviour of an SDS
  - Basic steps in building an SDS
  - State based SDS
  - VoiceXML

Thank you!

Filip Jurčíček

Institute of Formal and Applied Linguistics
Charles University in Prague
Czech Republic

Home page: http://ufal.mff.cuni.cz/~jurcicek