Intelligent Personal Assistants

Sebastian Springenberg
June 22, 2016

Speech Technology Seminar, Institute of Computer Science Hamburg
1. Introduction

2. Semantic Interpretation

3. Learning by Instruction Agent

4. Conclusion
Introduction
Motivation

What do you expect from an intelligent personal assistant?
Say what you want, let the system infer the best course of action.

- Good **speech recognition**, **speech synthesis** and **semantic interpretation**
- Good interaction via **dialog management**
- **Personalised** and **context aware** [2]
“Schedule a meeting with John Monday at 2pm.”

- **Recognize** user’s **intend** to create a meeting
- **Act** on this intend. Create calendar entry, write email to John...
- **Robustness:** Deal with ambiguities (which John? which Monday?...)
Intelligent Personal Assistant - Interaction Model

- Elicitation
- Speech Synthesis
- Clarification
- Question
- Dialog Management
- Missing Element(s)
- Interaction Context
- Natural Language Understanding
- Semantic Interpretation
- Complete?
- N
- Y
- World Knowledge
- Action Selection
- Inferred User Intent
- Output to User
- Speech Synthesis
- Best Outcome
Semantic Interpretation
Statistical Framework

• Data driven approach on semantic interpretation
  POMDP:
  • Maintain a system of beliefs
  → Update beliefs using Bayesian inference underlying a policy
  • Optimize policy using reinforcement learning
  → Learn statistical distributions via observations, infer posterior
Semantic Interpretation - Statistical Framework

Statistical Framework - POMDPs

• Problems:
  - Complex internal state: user’s goal + user’s input (significant uncertainty) + dialogue history
  - Complex mapping from dialogue states to possibly large action space
  → POMDPs usually involve a lot of approximations (rank and prune state values, invoke independence assumptions, summary state space...)
Semantic Interpretation - Statistical Framework

[Diagram of state transitions involving goal, input, history, and event nodes at times t and t+1]
Rule-based Framework

• Relies on inference engine operating on a knowledge base
• Traditionally domain specific with a complex architecture of many components

Active Platform:
• More light-weight and developer friendly
• Loosely coupled services with specialised task representations

→ Active ontology for every task
Dynamic processing in a relational network of concepts

- Concepts allow various instantiations of canonical objects
- “Monday at 2pm”, “tomorrow morning”
  → date(DAY, MONTH, YEAR, HOURS, MINUTES)
- Optional, mandatory, unique and multiple children nodes
Which problems might arise when relying solely on a rule-based (active) platform?
Learning by Instruction Agent
• Commercial systems (Google Now, Siri, Cortana...) limited to predefined commands
• Learn new commands from natural language instructions
• Learning by instruction agent (LIA) [1]
• Teach agent how to achieve commands through sequence of steps
• Lexicon Induction: Ability to generalise across taught commands:
  “Forward e-mail to Lisa.” → “Forward email to Ben.”
“I’m stuck in traffic and will be late.”

Instructions:

• “First, use GPRs to estimate time of arrival.”
• “See who I am meeting.”
• “Send an email to this person indicating that I’ll be late.”

→ System now understands how to handle similar situations in the future.

• Also possible to teach agent the same action sequence for different commands:
  “Send an e-mail to Lisa.” = “Drop Lisa an e-mail”
LIA - Learning by Instruction Agent

LIA

- Operates in an e-mail domain. Actions: Read, Compose, Send...
- Interaction via a text dialogue.
- Semantic parser (assign semantics to commands): Map commands to logic form
- Back-end (execute commands): built-in executables + declarative knowledge base
Instructing LIA

1. Teach new **declarative** knowledge:
   - Define new concepts along with fields and instances
   - “contact” → “has an email address”, ”Lisa is a contact”

2. Teach new **procedural** knowledge:
   - How to execute a new command
   - Which already known executables to use to achieve the task
Semantic Parser

- Combinatory Categorial Grammar (CCG) parser: Words behave like functions

1. Lexicon
   - Table mapping words to syntactic categories and logical forms
   - Syntactic category: how entry can combine with other words

2. Set of grammar rules
   - Decompose phrases via function operations (application, composition...)

3. Trained parameter vector
   - Multiple parses possible, train parser to select best solution
LIA - Learning by Instruction Agent

Back-end Command Executer

• Evaluate logical forms
• **Primitive executable** functions predefined
  
  `sendEmail, addFieldToConcept, createInstance`
  
• Important **functions for learning** new commands:
  
  `unknownCommand, teachNewCommand`
Learning new Commands

unknownCommand

semantic parser

lexicon induction

single logical form doSeq(...)

instruction interaction teachNewCommand

sequence of logical forms

NLI
Lexicon Induction

• Given the instructed command, update semantic parser to **generalise**

• Find **subexpressions**
  Spans which construct logical forms that are part of the complete logical form

• Declare subexpressions as **arguments** that can be filled during parsing

• “forward to Lisa” → “Lisa” parses to *lisa* → subexpression and thus argument to “forward to”
## Semantic Parser training examples

<table>
<thead>
<tr>
<th>Text Command</th>
<th>Logical Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>set the subject to time to go</td>
<td>(setFieldFromString (getMutableFieldByFieldName subject) (stringValue &quot;time to go&quot;))</td>
</tr>
<tr>
<td>send the email</td>
<td>(send email)</td>
</tr>
<tr>
<td>set body to email’s body and send email</td>
<td>(doSeq (setFieldFromStringVal (getMutableFieldByFieldName body)</td>
</tr>
<tr>
<td></td>
<td>(evalField (getFieldByInstanceNameAndFieldName email body))) (send email))</td>
</tr>
<tr>
<td>add length as a field in table</td>
<td>(addFieldToConcept table (stringNoun &quot;length&quot;))</td>
</tr>
<tr>
<td>forward to charlie</td>
<td>(doSeq (doSeq (doSeq (createInstanceByConceptName outgoingemail)</td>
</tr>
<tr>
<td></td>
<td>(setFieldFromStringVal (getMutableFieldByFieldName subject) (evalField</td>
</tr>
<tr>
<td></td>
<td>(getFieldByInstanceNameAndFieldName email subject))) (setFieldFromStringVal</td>
</tr>
<tr>
<td></td>
<td>(getMutableFieldByFieldName body) (evalField (getFieldByInstanceNameAndFieldName</td>
</tr>
<tr>
<td></td>
<td>email body))) (setFieldFromStringVal (getMutableFieldByFieldName recipient)</td>
</tr>
<tr>
<td></td>
<td>(evalField (getFieldByInstanceNameAndFieldName charlie email))) (sendEmail))</td>
</tr>
</tbody>
</table>

[1]
## Lexicon entries

<table>
<thead>
<tr>
<th>Word</th>
<th>Syntactic Category</th>
<th>Logical Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>set</td>
<td>((S/PP.StringV)/MutableField)</td>
<td>(lambda x y (setFieldFromString x y))</td>
</tr>
<tr>
<td>to</td>
<td>PP.StringV/StringV</td>
<td>(lambda x x)</td>
</tr>
<tr>
<td>subject</td>
<td>FieldName</td>
<td>subject</td>
</tr>
<tr>
<td>send</td>
<td>S/InstanceName</td>
<td>(lambda x (send x))</td>
</tr>
<tr>
<td>email</td>
<td>InstanceName</td>
<td>email</td>
</tr>
<tr>
<td>set</td>
<td>((S/PP.FieldVal)/MutableField)</td>
<td>(lambda x y (setFieldFromFieldVal x y))</td>
</tr>
<tr>
<td>to</td>
<td>PP.FieldVal/FieldVal</td>
<td>(lambda x x)</td>
</tr>
<tr>
<td>and</td>
<td>(S/S)\S</td>
<td>(lambda x y (doSeq x y))</td>
</tr>
<tr>
<td>'s</td>
<td>((Field\InstanceName)/FieldName)</td>
<td>(lambda x y (getFieldByInstanceNameAndFieldName y x))</td>
</tr>
<tr>
<td>forward</td>
<td>S/InstanceName</td>
<td>(lambda x (doSeq (doSeq (doSeq (createInstanceByConceptName outgoingemail) (setFieldFromFieldVal (getMutableFieldByFieldName subject) (evalField (getFieldByInstanceNameAndFieldName email subject))))) (setFieldFromFieldVal (getMutableFieldByFieldName body) (evalField (getFieldByInstanceNameAndFieldName email body))) (setFieldFromFieldVal (getMutableFieldByFieldName recipient) (evalField (getFieldByInstanceNameAndFieldName x email)))) (sendEmail)))</td>
</tr>
</tbody>
</table>
Conclusion
Conclusion

• The development of an intelligent personal assistant involves several complex tasks: speech recognition, speech synthesis, dialogue management, semantic interpretation...

• Understanding the user’s intent plays a crucial role.

• Both statistical and rule-based semantic interpretation reveal benefits and drawbacks.

• Instructable agents can help rule-based systems to improve on problems related to predefined commands and make the system more flexible.
A. Azaria, J. Krishnamurthy, and T. M. Mitchell. 
*Instructable Intelligent Personal Agent.* 
2016.

W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. 
*Listen, attend and spell.*

*Natural Interaction with Robots, Knowbots and Smartphones.*