Programmable Prefrontal Cortex

- Planning: Developer code in Geniescript
- Moderated social behavior: control + refining to LLM
- Truthfulness: Finetuned LLM to create DB queries
Agent Anatomy

1. **Contextual Semantic Parser:**
   (Dialogue State Tracking DST)
   User utterance + context $\rightarrow$ user state

2. **ThingTalk Execution:**
   ThingTalk $\rightarrow$ result state

3. **Agent policy:** code to decide the response
   User context + result $\rightarrow$ agent state

**Natural Language Generator (NLG):**
agent state $\rightarrow$ agent utterance

4. **Output Context:**
   Summarizes the dialogue to date
   Becomes input context for next turn
Architectures of Virtual Assistants

• Grounded LLMs: Task-Oriented Virtual assistant
  • Classical: User-initiatives only – add conversational [Example]
  • Guiding: Keeps the conversation going [Alexa, Siri]
  • Proactive: Actively offers advice [Restaurant]
  • Instructive: Leads the conversation; occasion user initiatives [Facilities Management]

• + LLMs
  • Task-oriented assistant improved with LLM [Wiki-Genie]
  • Constrained interaction with social intelligence: [Autism coach]
  • Open-domain chitchat with trust: [Fact-checked LLM]
  • Chat with memory [Persona-Based Chat]
Programming Virtual Assistants

• GenieScript addresses the complexity in virtual assistant programming

• Contextual natural language understanding (Lecture 5)
  • Grounding LLMs with external knowledge bases + APIs
    • Contextual semantic parsing
    • Training data synthesis (Lectures 7,8)
  • ConvoThingTalk: Conversation semantics (core sentence + Apply)
  • AutoNav: Automatic handling of knowledge base navigation

• Agent Initiatives
  • Common state between user and agent: Skills
  • Semantics of agent utterances → contextual semantic parser
  • Interaction between developer state machine and AutoNav

• Mixed Initiatives
  • Passing the floor between the user and developer
  • User interrupting the agent
Classical Assistant: Restaurant Reservation in Geniescript

Com.yelp Restaurant Skills

Knowledge Base (KB)

Restaurant [“restaurant”, “food place”]
id: Entity(Restaurant)
geo: Location
  [“address”, “in #”, “near #”, “around #”]
price: Enum(cheap, moderate, expensive)
  [“#-ly priced “, “#”]
cuisines: Array(Entity(Cuisine))
  [“# food”, “serves # food”]

Action

MakeReservation [“reserve #”, “book #”]
restaurant: Entity(Restaurant)
book_people: Number [min=1]
  [“for #”, “for # people”]
book_day: Date [“for #”]
book_time: Time [“at #”, “for #”]

main () {
  import com.yelp;
  dlg.say
    (“Hello, I am Genie, your restaurant assistant.”);
}

GenieScript: JavaScript + dlg package
**Example Dialogue for a Virtual Assistant in Genie**

<table>
<thead>
<tr>
<th>USER</th>
<th>GENIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello, I am Genie, your restaurant assistant.</td>
<td>I’d recommend Chez TJ in Mountain View. It has one Michelin star!</td>
</tr>
<tr>
<td>I’d like a French Restaurant.</td>
<td>The distance to Chez TJ is 7.5 miles.</td>
</tr>
<tr>
<td>The distance to Left Bank is 2.8 miles.</td>
<td>Which day would you like to book Left Bank?</td>
</tr>
<tr>
<td>Can you recommend something nearby?</td>
<td>The price is moderate.</td>
</tr>
<tr>
<td>Perfect. Please book the restaurant</td>
<td>OK</td>
</tr>
<tr>
<td>Is it expensive?</td>
<td>Which day would you like to book Left Bank?</td>
</tr>
<tr>
<td>OK</td>
<td>November 2, 7pm, for 2 people.</td>
</tr>
<tr>
<td>Which day would you like to book Left Bank?</td>
<td>Could you confirm that you want a table for 2 people at Left Bank on November 2 at 7pm?</td>
</tr>
<tr>
<td>Yes.</td>
<td>I booked Left Bank for 2 on November 2 at 7pm.</td>
</tr>
<tr>
<td>What is the reservation number?</td>
<td>10382345.</td>
</tr>
</tbody>
</table>

Imagine writing this in a dialogue tree for each domain!
AutoNav Abstract Dialogue States

Any Time

KB Navigation

Action

Cancel

Query (TT)

SlotFillQuestion

QueryResponse

Done

InvokeAction (TT)

SlotFillQuestion

AsktoConfirm

ActionQuery (TT)

ActionQueryResponse

ActionResponse

Done

requested

confirmed

User

Agent
Classical Assistant Example 2: Wiki-Genie in geniescript

```java
main () {
    import org.wiki;
    dlg.say
        ("Hello, I am Genie, your restaurant assistant.");
}
```
Quiz:

• What kind of training data do we need for the semantic parser?
Architectures of Virtual Assistants

- Grounded LLMs: Task-Oriented Virtual assistant
  - Classical: User-initiatives only – add conversational
  - Guiding: Keeps the conversation going
  - Proactive: Actively offers advice
  - Instructive: Leads the conversation; occasion user initiatives

- + LLMs
  - Task-oriented assistant improved with LLM
  - Constrained interaction with social intelligence:
  - Open-domain chitchat with trust:
  - Chat with memory

[Example]
[Alexa, Siri]
[Restaurant]
[Facilities Management]
[Cooking]
[Wiki-Genie]
[Autism coach]
[Fact-checked LLM]
[Persona-Based Chat]
Guiding Assistant: Personalized Restaurant Assistant

Restaurant Skills

Knowledge Base (KB)

| Restaurant [“restaurant”, “food place”] |
| id: Entity(Restaurant) |
| geo: Location |
| price: Enum(cheap, moderate, expensive) |
| cuisines: Array(Entity(Cuisine)) |

Action

MakeReservation [“reserve #”, “book #”]
| restaurant: Entity(Restaurant) |
| book_people: Number [min=1] |
| book_day: Date [“for #”] |
| book_time: Time [“at #”, “for #”] |

Code

```geniescript
main () {
    dlg.say ("Hello, I am Genie.");
    // User profile: last had pizzas 20 days ago
    result = dlg.initiateQuery ("Find a restaurant that serves pizza",
                               @restaurant(), filter cuisine==pizza"
                           "You haven't had pizza for a while, would you like that?");
    if (result.status == “success”) {
        dlg.initiateAction ("Book ‘result1.id’",
                            @restaurant(), @restaurant='result1.id'
                        "Would you like me to book ‘result1.name’?"
                  );
        dlg.say ("Goodbye.");
    }
}
```
Example 1: Dialogue

USER
Hello, I am Genie. Are you interested in some pizzas today?
Sure.
Sounds good. I heard it is good. (Done)
OK
November 2, 7pm, for 2 people.
Yes.

GENIE
I’d recommend Terun in Palo Alto.
Would you like me to book Terun?
Which day?
Could you confirm that you want a table for 2 people at Terun on November 2 at 7pm?
I booked Terun for 2 on November 2 at 7pm.
Example 2: Dialogue

USER
Hello, I am Genie. Are you interested in some pizzas today?
I'd recommend Chez TJ in Mountain View. It has one Michelin star!
How far is it from here?

GENIE
The distance to Chez TJ is 7.5 miles.
Can you recommend something nearby?
The distance to Left Bank is 2.8 miles.
Perfect. (Done)

USER
Is it expensive?
The price is moderate.
OK
Sure.

GENIE
Would you like me to book Left Bank?

USER
November 2, 7pm, for 2 people.

GENIE
Which day would you like to book Left Bank?

USER
Could you confirm that you want a table for 2 people at Left Bank on November 2 at 7pm?

GENIE
Yes.
I booked Left Bank for 2 on November 2 at 7pm.
Guiding Assistant: Abstract Dialogue States

**KB Navigation**
- **Cancel**
- **Query (TT)**
- **InvokeAction (TT)**

Any Time

- **InitiateQuery**
- **Query (TT)**
  - **SlotFillQuestion**
  - **QueryResponse**
- **Done**

**Action**
- **InitiateAction**
  - **InvokeAction (TT)**
  - **SlotFillQuestion**
  - **AsktoConfirm**
- **ActionQuery (TT)**
  - **ActionResponse**
- **ActionQueryResponse**
  - **Done**

User:
- **Agent-Response**
- **Agent-Initiative**
Dlg Primitives

• InitiateQuery (agent utterance) // ThingTalk meaning
• InitiateAction (agent utterance) // ThingTalk meaning
• Quiz:
  • What happens if the user ignores the agent’s query/action?
  • When should the agent get the floor? (when we invoke InitiateQuery)
  • What are other examples for agent initiatives?
  • Why do need the ThingTalk statement?
  • What kind of training data do we need?
  • What else might we need to program agent initiatives?
Other dlg Functions

• dlg.say (statement);
• dlg.expect (type, statement);

Types
• ThingTalk representation
• String (e.g. slotfill on a string parameter: “What is the message to send?”)
• Enumerated types
  • Yes, no: “Would you like to take a survey after the call?”
  • Small number of classes:
    “Is this statement positive, neutral, or negative

Quiz: how should we parse enumerated types?
Architectures of Virtual Assistants

- Grounded LLMs: Task-Oriented Virtual assistant
  - Classical: User-initiatives only – add conversational [Example] [Alexa, Siri]
  - Guiding: Keeps the conversation going [Restaurant]
  - Proactive: Actively offers advice [Facilities Management]
  - Instructive: Leads the conversation; occasion user initiatives [Cooking]

- + LLMs
  - Task-oriented assistant improved with LLM [Wiki-Genie]
  - Constrained interaction with social intelligence: [Autism coach]
  - Open-domain chitchat with trust: [Fact-checked LLM]
  - Chat with memory [Persona-Based Chat]
Proactive Assistants

• Offers advice to users continuous, possibly at every turn
• Useful for novices
  • They may not know how to do their job well
  • They need help discovering the features of the Assistant
• Also useful for experts
  • Save them time on common next steps
Facility Management MultiModal Assistant
Junwen Zheng, CIFE, Q4 2021, Q1 2022

Construction industry: 2\textsuperscript{nd} least digitized industry
Digital information inaccessible to blue collar workers
  • Not tech savvy
  • Hands are busy
  • On the job, not at the desk
Voice interface: access info on the job naturally
Built with 100 manually annotated commands + synthesized data
for Facilities Management
**Example: Dialogue of a More Pro-Active Assistant**

<table>
<thead>
<tr>
<th>USER</th>
<th>GENIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello, I am Genie, your Facilities Management assistant</td>
<td>It is ACU 11415</td>
</tr>
<tr>
<td>What is the ID of the air conditioner that services the 3rd floor?</td>
<td>Would you like to see the manual of ACU 11415?</td>
</tr>
<tr>
<td>No, show me where it is.</td>
<td></td>
</tr>
<tr>
<td>(Visualize the 3rd floor of the building and where the ACU is)</td>
<td>Would you like to visualize the subsystem of ACU 11415?</td>
</tr>
<tr>
<td>Yes.</td>
<td>(Visualize the building system SA 15 that contains ACU 11415)</td>
</tr>
</tbody>
</table>

- **Question about an object** → **Show the manual**
- **Visualize an object** → **Isolate the subsystem**
A Proactive Multimodal Assistant in GenieScript

```java
main () {
    import org.fm;
    mode = eager;
    dlg.say('Hi! I am a Genie, a Facilities Management assistant')
    result = dlg.expect('org.fm', '"");
    while (true) {
        visualize(result);
        case (result.QueryType) {
            bldg_object:
                ID=result.ID;
                result = dlg.initiateQuery
                    ("Get manual of 'ID' ",
                     // "[manual] of @org.fm. bldg_object(), ID= 'ID'"
                     "Would you like to see the user manual of 'ID'?")
                break;
            forge_model:
                ID=result.ID;
                result = dlg.initiateAction
                    ("Visualize 'id' ",
                     // "@org.fm.forgemodel(), ID= 'ID'"
                     "Would you like to visualize the subsystem of 'ID'"");
                break;
        }
    result = dlg.expect('org.fm', '"");
}
```
Proactive Assistant: Abstract Dialogue States

Any Time

KB Navigation

- InitiateQuery
  - Query (TT)
  - SlotFillQuestion
  - QueryResponse
  - Done

Action

- InitiateAction
  - requested
  - InvokeAction (TT)
  - SlotFillQuestion
  - AsktoConfirm
    - confirmed
    - InvokeAction (TT)
    - ActionQuery (TT)
    - ActionResponse
    - ActionQueryResponse
    - Done
Quiz

• Can you suggest more examples for proactive assistants?
Architectures of Virtual Assistants

• Grounded LLMs: Task-Oriented Virtual assistant
  • Classical: User-initiatives only – add conversational
  • Guiding: Keeps conversation going
  • Proactive: Actively offers advice
  • Instructive: Leads the conversation; occasion user initiatives

• + LLMs
  • Constrained interaction with social intelligence:
  • Open-domain chitchat with trust:
  • Persona-based chitchat:

[Example] [Alexa, Siri] [Restaurant] [Facilities Management] [Cooking] [Autism coach] [Fact-checked LLM] [LLMs with memory]
Instructive Assistant: Cooking

import com.recipe, com.unitConversation, 

dlg.say ('Hi! I am your cooking assistant, at your service')
result = dlg.initiateQuery('find a recipe',
                           'What dish would you like to cook today?')

if (dlg.result == completed) {
    currentIdx = 1
    while (currentIdx < maxIndex) {
        dlg.executeAndRespond('what is the current step?')
        dlg.expect ('$DONE')
    }
    dlg.say ('bye')
}
Instructive Assistant

• The assistant runs a full program
  • Including executing skills
• Note that skills become a shared resource that can be invoked by both the user and the agent. (Multi-threading)

• Quiz: What if:
  • The user says: “how many teaspoons are in a tablespoon?”
  • The user says: “Go back to step 3”
Going Beyond DB Navigation

• Execution need not be database lookups
  • E.g. Facilities management: a multimodal assistant (Autocad Forge)
• ThingTalk can be extended
  • Access control:

```
Mom or Dad can
monitor my security camera
only if I am not at home

(source = “Dad” || source = “Mom”),
executor = SELF :
monitor @security_camera.event,
@phone.get_gps() {location≠home} ⇒ return
```

ThingTalk can trivially be mapped to SMT (Satisfiability Modulo Theory).
Theorem prover is used to enforce access control

• Next example: Web operations
Automated Call Agents

Research Question:
Can we train one universal agent?

That reads instructions on online help centers

Create an Account
1. Select the Account button in the upper-right corner of Walmart.com
2. Click Create an Account
3. Enter your name and email address on the sign-up screen
4. Create a password and confirm it
5. Select Create Account

Redeem a Gift Card
1. Find the claim code.
2. Go to Redeem a Gift Card.
3. Enter your claim code and select Apply to Your Balance.

Find your AD Account
0. Click Ads to open Ads Manager
0. Click Reporting
0. Click the Ads tab under the graph
0. Scroll down to the list of ads
0. The 12-digit number beginning with 687 under the ad name is the ad ID
RUSS
Rapid Universal Support Service
Nancy Xu, Sam Masling, Michael Du, Q4, 2020

- Trained once with synthesized data
- Extended ThingTalk for web operations
- Experiment
  - 22 different online help centers
  - 80 tasks, 741 instructions
  - Overall accuracy: 77%
  - Semantic parsing accuracy: 87%
  - Users prefer RUSS over the web

Grounding Open-Domain Instructions to Automate Web Support Tasks
Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, Monica S Lam
Going Beyond DB Navigation

Quiz: Other suggestions?
Architectures of Virtual Assistants

• Grounded LLMs: Task-Oriented Virtual assistant
  • Classical: User-initiatives only – add conversational
  • Guiding: Keeps conversation going
  • Proactive: Actively offers advice
  • Instructive: Leads the conversation; occasion user initiatives
  • + LLMs

• Constrained interaction with social intelligence:
• Open-domain chitchat with trust:
• Persona-based chitchat:

[Example] [Alexa, Siri] [Restaurant] [Facilities Management] [Cooking] [Autism coach] [Fact-checked LLM] [LLMs with memory]
I’m here to help hone your social skills. Let’s begin. Suppose I say “Tomorrow is supposed to be warm.”
Is this a positive, neutral, or negative statement?

Not quite. It is a neutral statement. How would you respond to that statement?

Good reply. You sound like you’re joking around and that you’re excited for the weather.
main () {
    dlg.say ("Hello, I am Noora, and I'm here to help hone your social skills. Let's begin.");
    for (i = 0; i < scenarios.length; i++) {
        dlg.say ("Suppose I say 'scenarios[i].string'");
        const response = yield * dlg.expect ("positive | neutral | negative",
            "Is this a positive, neutral, or negative statement?");
        if (response.result.sentiment == scenarios[i].sentiment)
            dlg.say ("That's right.") else dlg.say ("Not quite. It is a 'scenarios[i].sentiment' statement.");
        response = yield * dlg.expect (string, "How would you respond to that statement?");
        dlg.say (llm.gen (AddPromptforSentiment(scenarios[i].sentiment, response.result)));
    }
}
Chitchat Agent in GenieScript

main () {

    greeting = "Hello, Let's chat."
    dlg.say (greeting);
    history = greeting;
    while (true) {
        u = yield * dlg.expect (string);
        history = history + u;
        dlg.say (llm.gen (history));
    }
}
Fact-checked Chitchat Agent in **geniescript**

```java
main () {
    import org.wikidata;
    greeting = “Hello, Let's chat.”;
    dlg.say (greeting);
    history = greeting;
    while (true) {
        u = yield * dlg.expect (string);
        history = history + u;
        dlg.say (dlg.factChecked (llm.gen (history)));
    }
}
```
Fact-Checking

```java
factChecked (stmt: string) {
    s = stmt;
    while (true) {
        questions = llm.gen (AddPromptToGenerateQuestions (s));
        answers = dlg.parseExecute (questions);
        if (confirmed (answers)) break;
        s = llm.gen (AddPromptToReviseBasedonAnswers (s, answers));
    }
    return s;
}
```
Persona-based ChitChat in geniescript

// Goal: To talk to users based on their favorite movie stars
main () {
    greeting = “Hello, I love movies. Let’s chat. I love Brad Pitt. How about you?”;
    dlg.say (greeting);
    history = greeting;
    favActors = {};
    while (true) {
        u = yield * dlg.expect (string);
        newActors = llm.gen ( AddPromptToSummarizeActors (u) );
        actors = actors + newActors;
        history = history + u;
        response = llm.gen ( AddPromptToMentionActors (actors, history));
        response = genie.factChecked (response);
        dlg.say (response);
    }
}
Programming Virtual Assistants

• GenieScript addresses the complexity in virtual assistant programming
• Contextual natural language understanding (Lecture 5)
  • Grounding LLMs with external knowledge bases + APIs
    • Contextual semantic parsing
      • Training data synthesis (Lectures 7,8)
    • ConvoThingTalk: Conversation semantics (core sentence + Apply)
    • AutoNav: Automatic handling of knowledge base navigation

• Agent Initiatives
  • Common state between user and agent: Skills
  • Semantics of agent utterances → contextual semantic parser
  • Interaction between developer state machine and AutoNav

• Mixed Initiatives
  • Passing the floor between the user and developer
  • User interrupting the agent