CS 7637 Alternative Project 3 - Chatbot
Part 2
by Soumya Gosukonda

Introduction
This report expands upon the design and further implementation of a conversational agent called "Labs Informer" as specified in Part 1 Report for this project. The Labs Informer is designed to answer questions a prospective or new student may have about Research Laboratories under the School of Interactive Computing(SIC)[1] at GeorgiaTech. These can include questions about which labs work on a specific field of interest, who is the associated faculty, what does a specific lab do and so on.

Under the section "**Labs Informer 3.2: Design**", I elaborate on the Knowledge-Based Artificial Intelligence (KBAI) concepts used in the design of this agent . Further learning capabilities that can be added to my agent are discussed under section "**Labs Informer 3.2: Potential for Learning**".

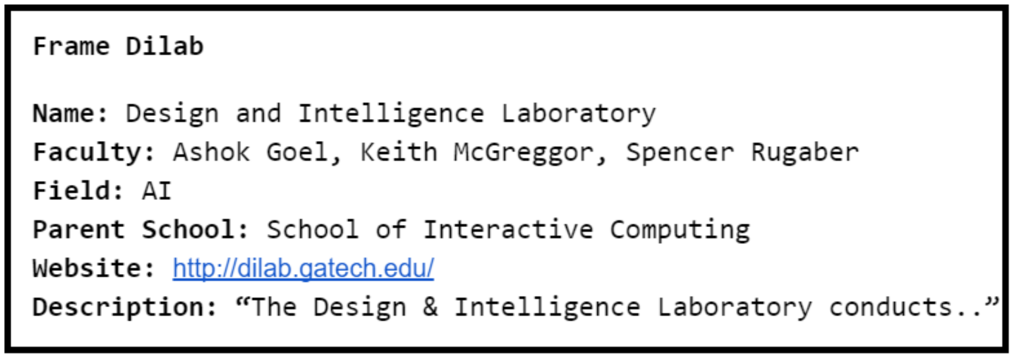
Labs Informer 3.2: Design
For Part 2 my goals were to enhance the agent's capabilities in the following aspects:

1. Expand user and agent querying capabilities
To widen the scope of conversation, I created a few more cases of possible user queries in addition to the ones in 3.1

The agent can now:
- Respond to a user greeting and ask for the user's name and use it in response.
- Query by a faculty name to retrieve the associated lab(s)
- Give improved responses in case it does not find the lab/faculty/field asked for.

In all these actions, the agent uses a combination of Frames[2], Classification[3] and Production Systems[4] primarily. The canonical forms of all entities[5] and their relationships are captured in Frames. Based on the user query, data from the relevant Frames is retrieved. A sample Frame for a Lab is shown in Figure 1. Figure 2 shows the relationship between Lab frame and Faculty frame.

[1] *School of Interactive Computing: http://www.ic.gatech.edu/*
[2] *Frames: https://en.wikipedia.org/wiki/Frame_(artificial_intelligence)*
[3] *Classification: https://en.wikipedia.org/wiki/Statistical_classification*
[4] *Production Systems: https://en.wikipedia.org/wiki/Production_system_(computer_science)*
[5] *Entities: https://docs.api.ai/docs/concept-entities*



```
Frame Dilab

Name: Design and Intelligence Laboratory
Faculty: Ashok Goel, Keith McGreggor, Spencer Rugaber
Field: AI
Parent School: School of Interactive Computing
Website: http://dilab.gatech.edu/
Description: “The Design & Intelligence Laboratory conducts..”
```

Figure 1. Frame for Dilab

To resolve a user query, several initial examples were provided to help define the possible user intent[6] and entities that the user may or may not mention. This helped in building a classifier that could classify already trained as well as some potentially similar user queries into their corresponding intent. For example, the classifier was trained on questions like: *“Could you tell me about Thad Starner?”*, *“how about dr ashok goel?”* etc. And upon asking a similar question not given in the training examples like *“What can you tell me about professor Balch?”*, the agent was able to respond with *“Tucker Balch heads the Computational Perception Lab”*. As mentioned in Part 1, this classifier has been implemented using API.ai[7] tools.

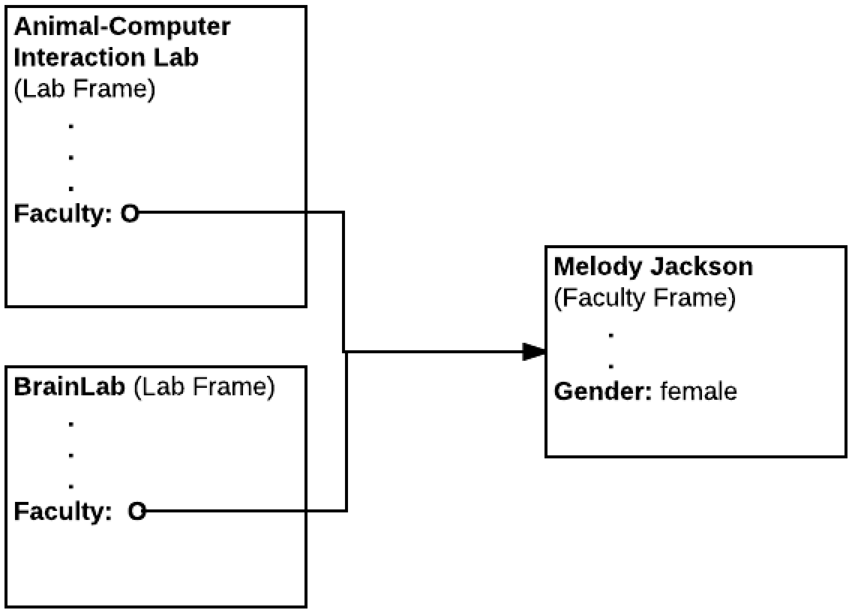

Animal-Computer Interaction Lab (Lab Frame)
Faculty:
BrainLab (Lab Frame)
Faculty:
Melody Jackson (Faculty Frame)
Gender: female

Figure 2. Relation between Lab Frames and Faculty Frame

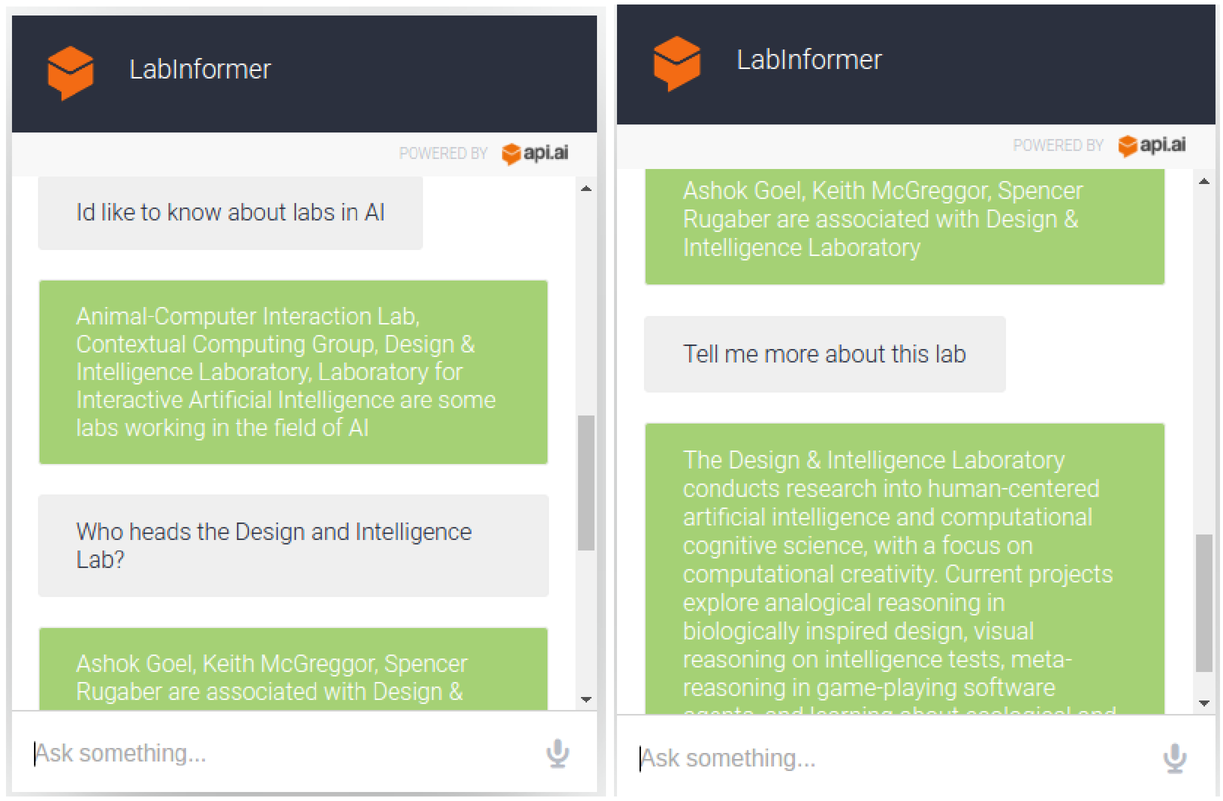
After classifying the user query into a specific intent, the agent conversationally extracts required parameters from user. For eg. if user asks - “Tell me about some labs.”, the agent prompts the user with - “What field can I interest you in?” to extract a field of user’s interest. Next, the user query is sent to the agent’s response generator[8] component. This component takes the canonical form of the parameter passed and retrieves relevant data based on the

[6] *Intents: https://docs.api.ai/docs/concept-intents*
[7] *API.AI: https://api.ai/*
[8] *Gosukonda, S., CS 7637 Alternative Project 3 - Chatbot Part 1, Section Chatbot : “Labs Informer”*

identified user intent. Production systems are used to generate a response based on retrieved data. The type of the parameters and retrieved data determine the response type and the retrieved data is fitted into the response appropriately. Cases for no data retrieved are also handled using production rules.

Handle spelling errors made by user
Labs Informer 3.2 handles spelling errors that the user may make. For any agent to succeed in conversation with a human, it has to accommodate errors in user input. Especially in case of spelling faculty member's names. Currently, the agent can identify the canonical form of user input entity up to a tolerance of 1 misspelt/added/deleted letter.

Maintain context between queries
In this version, the agent also maintains context during conversation, i.e., it remembers previous input allowing user to implicitly refer to entities being talked about.
An example is shown in Figure 3.



Id like to know about labs in AI

Animal-Computer Interaction Lab, Contextual Computing Group, Design & Intelligence Laboratory, Laboratory for Interactive Artificial Intelligence are some labs working in the field of AI

Who heads the Design and Intelligence Lab?

Ashok Goel, Keith McGreggor, Spencer Rugaber are associated with Design &

Ashok Goel, Keith McGreggor, Spencer Rugaber are associated with Design & Intelligence Laboratory

Tell me more about this lab

The Design & Intelligence Laboratory conducts research into human-centered artificial intelligence and computational cognitive science, with a focus on computational creativity. Current projects explore analogical reasoning in biologically inspired design, visual reasoning on intelligence tests, meta-reasoning in game-playing software

Figure 3. Sample conversation showing context

Notice how the user does not have to repeat the name of the lab to ask more details about it. Currently the context feature is implemented only for knowing more details about a lab.

Labs Informer 3.2: Potential for Learning

Human language is diverse and keeps evolving. For a chatbot to be useful, it needs to learn from its mistakes. Not only does language evolve, but so do the topics. In context of Labs Informer, information about labs, associated faculty and website links - all are subject to change with time. One way to stay updated is to have the database update from time to time.

A more efficient process would be if the agent could update its database by conversing with the user.
A simple method of learning can be implemented based on “Learning by Correcting Mistakes” methodology as a means of self repair. The agent watches out for mistakes, isolates it, validates the cause of mistake and then repairs it. For eg. currently, if the user asks *“Who is the faculty associated with Computational Perception Lab”*, the agent returns a list of faculty including *“Aaron Bobick”*, who is currently not at GeorgiaTech. Assuming the user knows this information, user may say - *“That is not correct”*. Phrases like *“not correct”* can trigger the mechanism to isolate a mistake the agent made. Agent may ask - *“What is not correct?”* and user can reply saying *“Aaron Bobick is no longer at GeorgiaTech”*. To validate whether the agent isolated the mistake, it can reconstruct the response without Aaron Bobick and see user’s reaction. The user would give a positive reaction, in this case, and the agent would repair its mistake by updating its database. In a different case, where multiple entities are being talked about or there are multiple contexts, then agent can cycle over the entities mentioned conversationally and isolate the mistake. Agent should also use a mechanism to generate the most probable error candidate so as to not irritate the user by wrongly identifying errors and seeking validation. One way to do this could be by storing history of recent conversations with other users and using information from those to generate a confidence number for its responses that seek validation.

A general algorithm for this can be designed as follows:

```
agent(user_input):
|   does user_input contain flags for error?
|   if (yes):
|   |   confirm error situation from user
|   |   entities <- fetch entities from context
|   |   for each entity (limited by max 3):
|   |   |   reconstruct response with modified entity property
|   |   |   reaction <- get user reaction to response
|   |   |   if (reaction is positive):
|   |   |   |   modify/repair database to reflect changes
|   |   |   else:
|   |   |   |   continue to next entity
|   |   if (repair performed)
|   |   |   learn repair strategy
|   |   |   resume conversation
|   |   else
|   |   |   learn repair strategy not effective
|   |   |   resume conversation
|   else:
|   |   construct appropriate response
end
```

References

1. Goel Ashok (2016). Frames (Video Lecture Slides). Retrieved from https://t-square.gatech.edu/access/content/group/gtc-ebf6-13dc-5243-86d1-676b47f99612/Video%20SlideDeck/07%20-%20Frames.pptx

2. Goel Ashok (2016). Production Systems (Video Lecture Slides). Retrieved from https://t-square.gatech.edu/access/content/group/gtc-ebf6-13dc-5243-86d1-676b47f99612/Video%20SlideDeck/06%20-%20ProductionSystems.pptx
3. Goel Ashok (2016). Classification (Video Lecture Slides). Retrieved from https://t-square.gatech.edu/access/content/group/gtc-ebf6-13dc-5243-86d1-676b47f99612/Video%20SlideDeck/11%20-%20Classification.pptx
4. Goel Ashok (2016). Learning by Correcting Mistakes (Video Lecture Slides). Retrieved from https://t-square.gatech.edu/access/content/group/gtc-ebf6-13dc-5243-86d1-676b47f99612/Video%20SlideDeck/23%20-%20Learning%20By%20Correcting%20Mistakes.pptx
5. API.AI - API & Docs. (2016). Retrieved from https://docs.api.ai/
6. Gosukonda Soumya (2016). CS 7637 Alternative Project 3 - Chatbot Part 1. Retrieved from https://drive.google.com/open?id=1Vt_mVYUlzUe9_NmSSQE01jJjHao8D9lTsD4dIOxGTw4