# DAT410 Module 7 Assignment 7 – Group 26

Yahui Wu (MPMOB) (15 hrs)
yahuiw@chalmers.se
Personal number: 000617-3918
Tianshuo Xiao (MPMOB) (15 hrs)
tianshuo@chalmers.se
Personal number: 000922-7950

March 7, 2023

We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part of other solutions

## Reading and reflection

This article describes the GUS framework-driven word system designed to support natural language interaction with users. GUS uses frameworks to represent knowledge about a specific domain, and it employs a semantic network to connect the frameworks together. The system also uses natural language understanding techniques to parse user input and map it to frames in the semantic network. There are some problems in natural dialog. The first is mixed initiative, which means when in a natural conversation between people, either clear expectations are not stated or not delivered at all. Next is indirect answers. It is not always clear what constitutes an answer to a question. Sometimes it only can make narrow expectations it has about the subject matter and the client's goals. The most important is sentence fragments. Discourse in natural conversation is by no means always a complete sentence. These conversations can almost invariably derive a rule from a question that converts a fragmentary response into a complete sentence expressing the same information.

In attention, there are some principles of program organization. We started with modularization, exploring tools and techniques for building and integrating independent modules. Language comprehension systems have to operate in a multiprocessor environment. In a system with many sources of knowledge and several independent processes, some part of the mechanism must usually be used simply to decide what should be done next. When the amount of data is large or complex, large files with formatted data need to be processed.

The components of GUS include a parser, a framework representation, and an inference engine. It uses a table lookup mechanism to find the appropriate template and generates the English by filling in the template form. But it is easy to infer from the conversation the mistaken notion that GUS contains solutions to far more problems than it does. Sometimes the answer does not satisfy the customer's needs because the customer wants additional needs outside of the motivation.

In conclusion, an intelligent language comprehended must have a high-quality parser, an inference component, and a well-structured knowledge database. Therefore, it is necessary to continue developing improved language analysis systems and knowledge representation languages.

## Implementation

```
import requests
import json
from geopy.geocoders import Nominatim

def get_weather(city):
    api_key = 'a05504ddaee060baff01c7de8a262bef'
    url = f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&
    units=metric'
    response = requests.get(url).json()
    temp = response['main']['temp']
    desc = response['weather'][0]['description']
    return f'The temperature in {city} is {temp} C  and the weather is {desc}.'

def get_pos(location):
    geolocator = Nominatim(user_agent="Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36")  #  ua
    location = geolocator.geocode(location)
    return location.latitude, location.longitude

def find_restaurant(location, cuisine):
    api_key = 'AIzaSyA97T9arklGc7qaXlsvqOOkiEBNcu9ipEY'
    url = f'https://maps.googleapis.com/maps/api/place/nearbysearch/json'
    lat, lon = get_pos(location)
    params = {'location':f'{lat},{lon}',
              'radius':'500',
              'type':'restaurant',
              'keyword':cuisine,
              'key':api_key}
    response = requests.get(url, params = params)
    outcome = response.json()['results']
    dic_res = {}
    for data in outcome:
        dic_res[data['name']] = [data['rating'], data['vicinity']]
    list_res = sorted(dic_res.items(), key = lambda x:x[1][0], reverse = True)
    return list_res

def get_stopid(stop):
    API_KEY = "a983c7e2-a956-38ab-8700-c95d69ea6eb2"
    LOCATION_NAME_URL = "https://api.vasttrafik.se/bin/rest.exe/v2/location.name"
    headers = {"Authorization": f"Bearer {API_KEY}"}
    params = {"input": stop, "format": "json"}
    response = requests.get(LOCATION_NAME_URL, headers=headers, params=params)
    response_data = json.loads(response.content)
    return response_data['LocationList']['StopLocation'][0]['id']

def trip(stop):
    stop = get_stopid(stop)
    # Set up the API endpoint and parameters
    url = "https://api.vasttrafik.se/bin/rest.exe/v2/departureBoard"
    API_KEY = "a983c7e2-a956-38ab-8700-c95d69ea6eb2"
    params = {
        "id": stop, # The stop ID
        "format": "json", # The desired response format
        "timeSpan": "15", # The time span to search for departures in (in minutes)
        "maxDeparturesPerLine": "1" # The maximum number of departures to return per
    line
    }
    headers = {"Authorization": f"Bearer {API_KEY}"}

    # Send the API request and retrieve the response
    response = requests.get(url, params=params, headers=headers)
```

```
    data = json.loads(response.text)

    # Extract the relevant information from the response
    departures = data["DepartureBoard"]["Departure"]
    trans = []
    outcome = list(departures[0].keys())
    cnt = 0
    for departure in departures:
        outcome = list(departures[cnt].keys())
        if 'rtDate' not in outcome:
            departures[cnt]['rtDate'] = 'unknown'
            departures[cnt]['rtTime'] = 'unknown'
        trans.append([departure["type"]+" "+"No."+departure["sname"], departure["
    direction"], departure["rtTime"]])
        cnt += 1
    return trans

while True:
    command = input('How can I help you? ').lower()

    if 'weather' in command:
        city = input('Which city would you like the weather for? ').lower()
        print(get_weather(city))

    elif 'restaurant' in command:
        location = input('Where are you located? ').lower()
        cuisine = input('What kind of cuisine are you in the mood for? ').lower()
        restaurants = find_restaurant(location, cuisine)
        print(f'I found {len(restaurants)} restaurants near {location}, the best 5
    with highest rating are:')
        cnt = 0
        for restaurant in restaurants:
            print(f'{restaurant[0]} with rating {restaurant[1][0]} at {restaurant
    [1][1]}')
            cnt +=1
            if cnt == 5:
                break

    elif 'transport' in command:
        stop = input('Which bus/tram stop are you going to? ').lower()
        trans_list = trip(stop)
        for tran in trans_list:
            print(f'{tran[0]} bound for {tran[1]} will arrive at {tran[2]}')
```

Listing 1: Dialogue systems and question answering

## Description of the system

We have created the following functions to perform our tasks.
The first function get_weather(city) takes a city name as input and uses the OpenWeatherMap API to retrieve the current temperature and weather description for that city. The function then returns a string that reports this information. We first retrieve the keyword 'weather' and then specific to a 'city', giving the corresponding temperature and weather conditions.
Then, we created a function that can get the latitude and longitude of the location. The function get_pos(location) takes a location name as input and uses the Geopy library to retrieve the latitude and longitude coordinates for that location. The function then returns these coordinates. This was to satisfy the parameters in the next function. Next, we created the find_restaurant(location, cuisine), which takes a location and cuisine type as inputs and uses the Google Places API to find nearby restaurants that match the cuisine type. The function returns a list of the top 5 restaurants sorted by rating.
The fourth function get_stopid(stop) takes a bus/tram stop name as input and uses the Västtrafik (Gothenburg Transportation System) API to retrieve the stop ID for that stop. The function then returns this ID. Then by using the function trip(stop) takes a bus/tram stop name as input and uses the Västtrafik API to retrieve the departure schedule for that stop. The function then returns a list of the upcoming departures for that stop, along with their type, destination and departure time.
Finally, we have designed interactive dialogues that rely primarily on keyword searches. If the command includes the word "weather", the code prompts the user for a city name and calls the get_weather() function. If the command includes the word "restaurant", the code prompts the user for a location and cuisine type and calls the find_restaurant() function. If the command includes the word "transport", the code prompts the user for a bus/tram stop name and calls the trip() function to retrieve the upcoming departures for that stop.

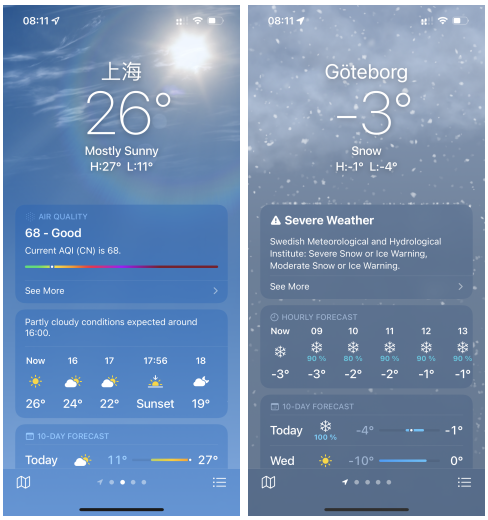
## Some sample outputs

In the following we will test the functionality of the dialogue system
**Weather forecast**
We asked the system about the weather in Gothenburg and Shanghai and it gave the following answer:

```
How can I help you? Can you tell me some weather information?
Which city would you like the weather for? Gothenburg
The temperature in gothenburg is -3.15° C and the weather is snow.
How can I help you? Can you tell me some weather information?
Which city would you like the weather for? Shanghai
The temperature in shanghai is 23.6° C and the weather is clear sky.
```

Figure 1: Weather

We compared the data given by Apple Weather and found that the conclusions given were basically correct.




Shanghai weather Gothenburg weather

**Restaurant recommendation**

We asked the system to recommend restaurants in Gothenburg, we chose to ask the system to recommend Chinese restaurants and Swedish restaurants and got the following results

```
What kind of cuisine are you in the mood for? Chinese
I found 8 restaurants near gothenburg, the best 5 with highest rating are:
Gansu Kitchen with rating 4.3 at Torggatan 9, Göteborg
Dubbel Dubbel 2 AB with rating 4.1 at Surbrunnsgatan 8, Göteborg
Wei Wei Asian Eatery with rating 3.9 at Postgatan 26, Göteborg
Ching Palace with rating 3.7 at Södra Hamngatan 2, Göteborg
Foodie Neo Asian Gourmet Kitchen (Göteborg) with rating 3.7 at Östra Hamngatan 5, Göteborg
```

Figure 2: Chinese restaurant

```
How can I help you? Do you have some recommended restaurants?
Where are you located? Gothenburg
What kind of cuisine are you in the mood for? Swedish
I found 20 restaurants near gothenburg, the best 5 with highest rating are:
Thörnströms Privata Rum with rating 4.7 at Postgatan 2, Göteborg
Bhoga with rating 4.7 at Norra Hamngatan 10, Göteborg
Södra Larm Bar & Bistro with rating 4.5 at Södra Larmgatan 9, Göteborg
Kåges Corner and Lunchbar with rating 4.5 at Stora Saluhallen, Göteborg
Löfqvist & Vi with rating 4.5 at Östra Hamngatan 40, Göteborg
```

Figure 3: Swedish restaurant

**Find the next tram/bus**

In our dialogue system we can provide some real-time information on station arrivals, but only in Gothenburg. Let's take Chalmers as an example

```
How can I help you? can you give me some stop transport?
Which bus/tram stop are you going to? chalmers
TRAM No. 13 bound for Sahlgrenska will arrive at 15:01
TRAM No. 7 bound for Tynnered will arrive at 15:03
TRAM No. 7 bound for Bergsjön will arrive at 15:01
BUS No. 753 bound for Mölndal via Åby, Påstigning fram will arrive at 15:01
TRAM No. 6 bound for Kortedala will arrive at 15:03
TRAM No. 6 bound for Länsmansgården via Sahlgrenska will arrive at 15:01
TRAM No. 10 bound for Biskopsgården will arrive at 15:01
BUS No. 16 bound for Fyrktorget will arrive at 15:03
TRAM No. 13 bound for Brämaregården will arrive at unknown
BUS No. 16 bound for Bockkranen will arrive at 15:02
TRAM No. 10 bound for Guldheden will arrive at 15:06
TRAM No. 8 bound for Angered will arrive at 15:07
TRAM No. 8 bound for Frölunda will arrive at 15:05
BUS No. 16 bound for Marklandsgatan will arrive at 15:05
BUS No. 258 bound for Brottkärr, Påstigning fram will arrive at 15:12
BUS No. 753 bound for Heden, Påstigning fram will arrive at 15:13
```

Figure 4: Gothenburg tram/bus information

**Limitations of our system**

1.Limited scope: The system is limited to certain geographic regions and APIs. For example, it can only find restaurants using the Google Places API and public transportation information using the Västtrafik API. Users in other regions may not be able to use the system effectively.

2.Limited natural language processing: The system relies on keyword detection to determine the user's intent. It may not be able to handle more complex user queries or understand natural language as well as a more sophisticated chatbot.

3.Lack of personalization: The system does not learn from user interactions or personalize responses based on the user's history or preferences.

# References


[1] Kamil, Czarnogorski. Monte Carlo Tree Search – beginners guide. Retrieved MAR, 24, 2018, from https://int8.io/monte-carlo-tree-search-beginners-guide

[2] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016). https://doi.org/10.1038/nature16961

# Reflection on the previous module

## Yahui Wu

In the previous module, we implemented a tic-tac-toe game playing system based on Monte-Carlo tree search (MCTS) . To extend our system and apply it to other areas rather than only playing games, we could build a recommendation system of a online video platform such as YouTube using MCTS. In the system, the root node represents a user's behaviour that this user starts to watch a video, and the state is the video watched by this user at the moment. The reward of selecting a certain node is the watch time of this video. The data needed for building such a system could come from the log stored by the server of the platform.
Besides the game playing system based on traditional MCTS methods, we discussed about more advanced game playing systems such as AlphaGo. AlphaGo combines MCTS and deep learning. It used human knowledge to learn preliminary versions of policies $p_\pi$ and $p_\sigma$. A next step was made based on the predicted move which is most likely to what an human expert would do at this state. Additionally, AlphaGo improved the human policies by using a new selection policy $p_\rho$ which was initialized at the very beginning and improved using reinforcement learning. However, using human policies has limitations for generalization and scaling of the data. As a more advanced version of the original AlphaGo, AlphaGo Zero learns only from self-play and a single network for both selection policy and evaluation. Besides, it uses only board positions as features and does not use "roll-outs" at all but only the evaluation network. Moreover, we learned something about open AI taking DoTA 2 as an example.

## Tianshuo Xiao

In the last assignment, we designed the game system using Monte Carlo tree search. But tree search also has some drawbacks. For example, it is not possible to explore all states and store the values of all states in the exploration, but machine learning can avoid these.
Monte Carlo algorithms can be applied not only to game systems, but also to other AI systems. In the diagnostic system, doctors and patients are applied to MCTS as our objects. The actions taken are testing and treatment, using the environment as anything related to the patient, observing the patient's disease symptoms so far. For the reward function, we can set the severity of the disease. MCTS can also be applied to recommender systems. In YouTube's recommendation video, we set the agent to the recommendation policy, the environment to the user's preferred video, the observation state to the user's viewing time and the time and frequency of skipping that video, and set the reward function to the total viewing time. In addition, we can also apply this to dialogue systems by setting the reward function to the number of interactions, in order to evaluate the interpretability of the dialogue system.
In Go the 1v1 matchmaking model is used, but in a 5v5 game such as Dota2, it faces problems such as long-time scopes, partial observability, and high-dimensional action/state spaces. Therefore, this requires deep learning, which shows the capabilities of current reinforcement learning.

# Summary of lectures

## Yahui Wu

Dialogue systems and question answering introduction
February 28, 2023
In the last class, we learned something about dialogue systems and question answering. Chatbots and digital assistants are applications of dialogue and question answering systems. Basically, a dialogue is a sequence of turns and in daily spoken dialogue, it is hard to find a end state of a turn. There are some important properties of dialogue such as speech acts, dialogue structure, grounding, sub-dialogue, clarification, initiative and inference and implicature. Speech acts consists of several types of acts which represents different speaker's attempts in dialogue. Constatives commits the speaker to something's being the case such as answering, claiming, confirming, denying. Directives are attempts by the speaker to get address to do something. Commissives commits the speaker to some future course of action. Acknowledgements express the speaker's attitude regarding the hearer with respects to some social action. At the beginning of dialogue, it will be initiated by the user or system or by both. Sometimes users won't put their point directly, the system needs to infer what users really want. Chatbots could be built on keyword-based which prefers to respond based on specific keyword. Corpus-based chatbots generate dialogue beased on very large datasets of real converstations. IR-based chatbots return response to most similar user turn or return most similar response. Dialogue manager is an important part of dialogue system architecture, it can be built on many types of systems described above. Frame-based or form-based dialogue manager is another option which classify dialogue to different forms that suits for different contexts or tasks. In each form the empty slots constrained to certain possible values and can be associated with questions users asked.

## Tianshuo Xiao

Dialogue systems and question answering introduction
February 28, 2023
In the last lecture, we learned about dialogue systems, the history of their development and their specific functional implementation.
Analyzing speech acts in conversations is an important part of the process. In a conversation, the person involved in the conversation needs to give a response or his or her own opinion according to the actual situation. In addition to this, it is necessary for the speaker to order the listener to make certain suggestions and actions, and to show an attitude towards certain behaviors of the listener.
Initiative in conversation encompasses the user, the system, and the mix. This entails determining whether neighboring requests are yes or no, correcting errors in subconversations, and responding to questions for clarification.
We go through the chatbot ELIZA as an example, a system based on keyword search, with preferred responses based on the most specific keywords. ELIZA can transform part of the syntax, for example, I'm – you are, my – you, you – me: what makes you think I xx you. Later developments of chatbots, including mental state models that influence conversations, were based on large datasets of real conversations and were able to respond based on a single response from the user's last turn. IR-based chatbots return response to most similar user turn and most similar response (based on word vector or word embedding).
The dialogue system architecture consists of the following components. First, speech recognition is processed, followed by natural language understanding (text-based dialogue). The processed language enters the Dialogue Manager, and the Task Manager classifies the tasks of the Dialogue Manager (commercial dialogue managers are mostly rule-based). Finally, natural language generation(mostly template-based) is implemented to synthesize text to speech.
In addition, dialogue management can be based on finite states, which are related to specific common

situations. In frame-based system, the understanding of words is related to the frame of reference of the psyche. A rich information state, beyond simple form filling, allows the system to move between multiple frames.