

# Conversational AI

[Arti Chauhan]

## Introduction

This assignment challenges us to design an Agent that can engage in a conversation like a human, as opposed to merely responding to the user's question. This paper leverages Knowledge-based AI concepts taught by Dr. Goel and Dr. Joyner so far, to design such an agent that will be capable of conversing on topics related to Georgia Tech OMSCS program.

## Background

Conversational AI has become an active field of research as Chatbots are becoming increasingly popular. Many companies are using Chatbots to promote their brand, connect with their users and cut operational costs in customer care centers. These bots are capable of handling pairwise utterance but fall short in longer conversational contexts and hence don't do very well connecting with the users at social and emotional level. Aim of this paper is to design a framework that would enable Agent to leverage 'content as well as context' to provide more meaningful conversation.

## Agent's architectural overview

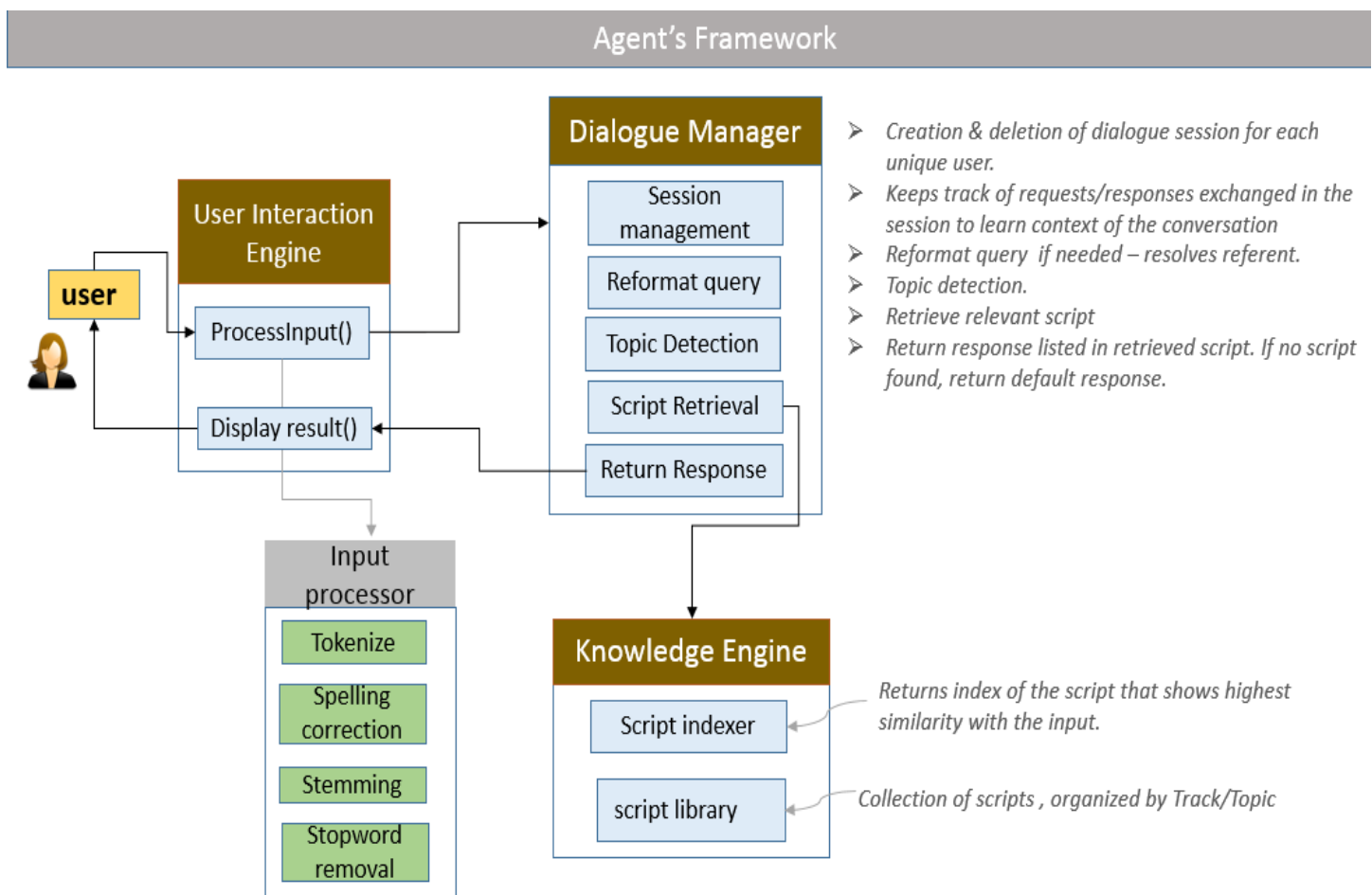


Figure 1:

## Work flow

Agent has three main modules – User Interaction Engine, Dialogue-Manager and Knowledge Engine.

1. *User Interaction module provides a natural language interface to its user. Conversational interface can range from a simple command-line to sophisticated GUI displaying other information that may be of interest. However, this paper assumes a simple chat CLI, but design is generic enough to take any form of text input (email, posts etc.).*
2. *User Interaction module calls its 'input-processor' sub-module which performs tokenization, spelling correction, stemming and stop word removal.*
3. *Processed input is then passed to Dialogue-Manager. Dialogue-Manager binds the user to a unique dialogue/session Id. This session keeps track of requests/responses exchanged with user, which enables Agent to learn context of the conversation, resolve referent and detect topic being discussed.*
4. *Dialogue-Manager invokes Knowledge Engine to retrieve relevant script and returns the response listed in retrieved script to User Interaction module, which then displays it to the user.*
5. *Knowledge Engine maintains a script repository, organized by Track/Topic. It finds the script that shows highest similarity to the input and returns it to Dialogue-Manager.*
6. *If no script is found, user is asked to clarify the question. If after N attempts, Agent is still not able to find a relevant script, a default response (e.g.: an apology message) is displayed to the user and Agent tries to make up for this shortcoming by suggesting a topic or presenting information that user might find helpful. Having picked some cues about the context during the course of conversation, Dialogue-Manager makes educated guesses on what topics to suggest or what information to present.*

## Knowledge representation (Q1-4)

For a computer program to be able to engage in a conversation with a human, it needs to be capable of understanding and processing natural language.

To emulate conversational capability akin to a human, Agent has to have

1. *Capability to understand voice or text (and perhaps visual) inputs. ( Although this paper will discuss design for text-based Agent only)*
2. *Memory to store past experiences (chats).*
3. *Capability to recall past experiences when it encounters a similar situation.*
4. *Capability to understand and infer from the context of the conversation.*

To achieve above-mentioned goals, agent will leverage case-based reasoning (CBR) framework, Semantic Nets and scripts.

- *For language translation and understanding agent will utilize Semantic Nets*
- *It will use scripts to store and understand conversational context.*
- *Agent will use CBR framework for script storage and retrieval.*

CBR is both a problem-solving methodology and a theory of reasoning and memory that is based on how humans reason from experience. In nutshell, a CBR system reasons by first retrieving a relevant prior case from its memory of cases , reusing or revising (adapting) the solution of the old case to solve the new problem and retaining the updated solution .

Since conversation is causally coherent (one utterance triggers another), scripts are more suited for this task as compared to Frames. While Frames tell the system where to look and what to look for a current situation, Scripts are bit more advanced data structure for knowledge representation as it tells the system what to do next, in addition to where and what to look for. Scripts can be thought of as a tree or network of states, driven by events. It defines rules, which control sequence of events to be executed.

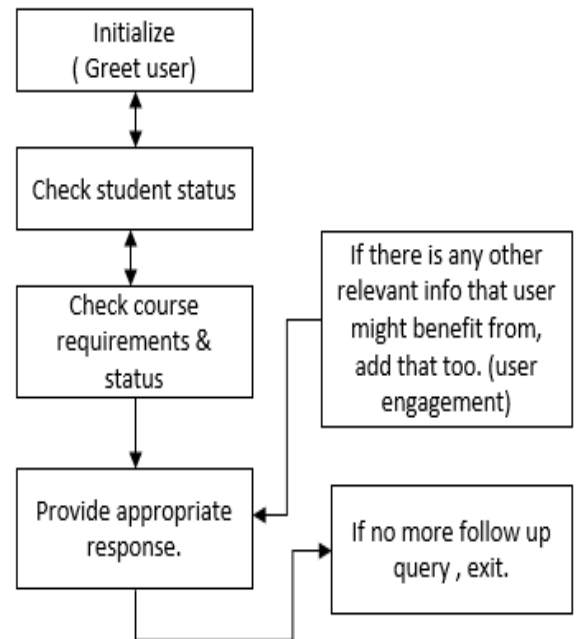
Text based Conversational Agents (CA) are typically organized into contexts consisting of a number for hierarchically organized rules. Each rule defines a structural pattern of sentences and corresponding response. One of the earliest text based CA 'ELIZA' relied on simple pattern matching techniques and mapping key terms of user input onto a suitable response. Due to unbounded nature of user input, a simple pattern matching techniques doesn't scale well. Instead, use of sentence similarity measures has been quite effective in reducing the scripts required [2]. For this purpose, Agent will be using python's NLTK package.

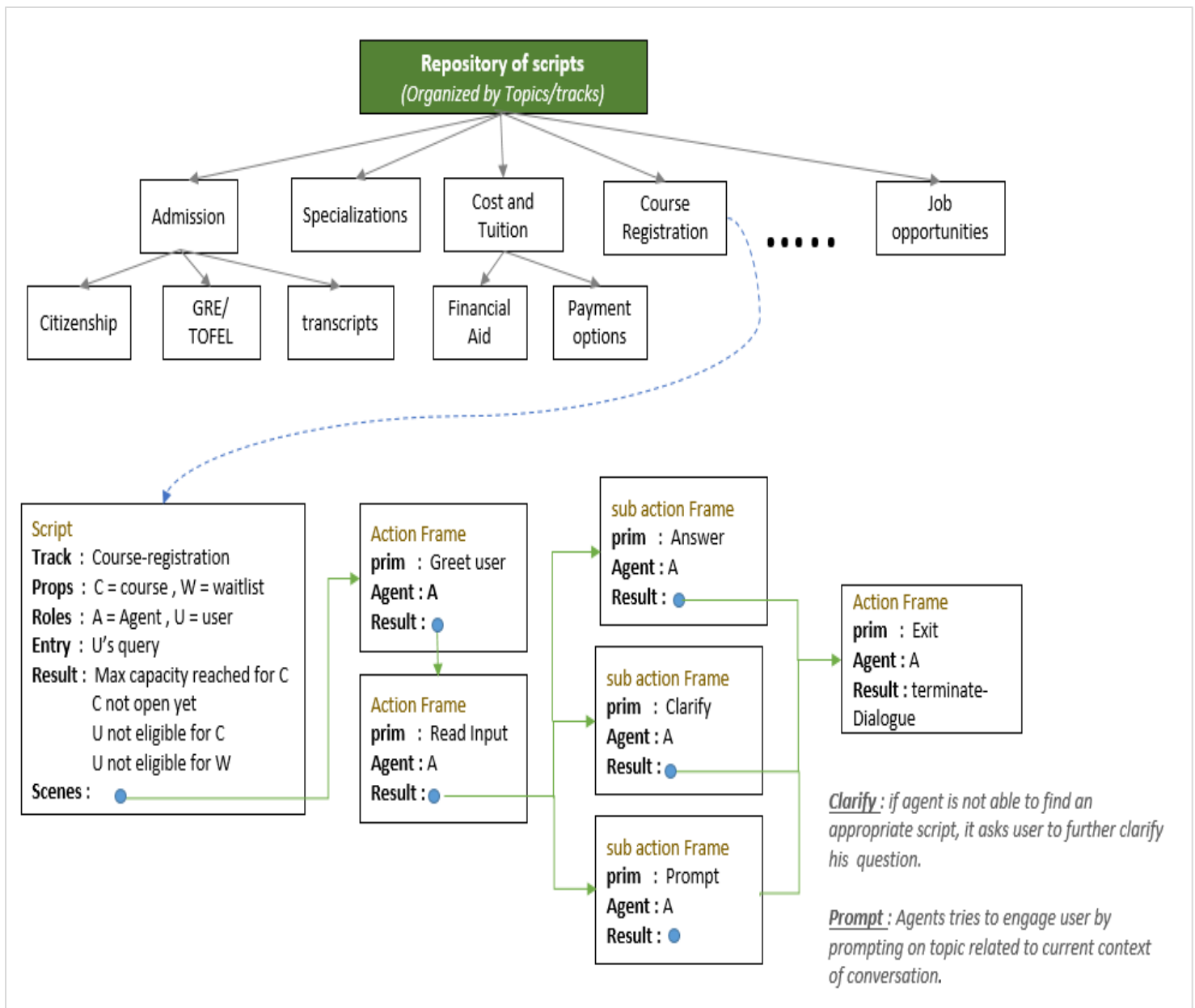
Below is a sample script for a conversation described on left. This script is invoked when user's query is related to registration issue. Though the conversation (on left) is specific to CS-7646, script on the right captures rules at an abstract level. Hence this script can be applied to registration issue for any course and not just for CS-7646.

### Sample case : Student chatting with Agent regarding course registration issue

Agent : Hi , How may I help you today ?  
User : I am not able to register for CS-7646.  
Agent : This course is a non-foundational course .Is this your 1<sup>st</sup> semester ?  
User: no, this is my 3<sup>rd</sup>.  
Agent :. Have you met your foundational course requirements ?  
User : yes I have.  
Agent : I am sorry you are running into this issue. I checked course status and it has reached it max capacity. But you may join waitlist.  
User : thanks. I will do so.  
Agent : You are welcome. Just so you know, waitlist status is regularly updated and you can check the status at link x. is there anything else I can help with ?  
User : good for now. Bye  
Agent : Bye

### A sample script for dialogue described on left





**Figure 2:** script organization and script structure

**Knowledge Acquisition:** Currently design assumes that Agent is trained on real conversations and have its script repository prebuilt in semi supervised manner. In other words, domain expert manually corrected the rules that were not learned correctly. This aspect of design is not scalable and should be enhanced in future for automatic script generation and adaptation.


## Input processing (Q6)

Agent provides a natural language interface to its user, which takes in text input in English. It calls 'input-processor' sub-module which performs tokenization, spelling correction, stemming and stop word removal.

1. Agent uses python's Natural language Took kit (NLTK) to perform these tasks , which provides functions for tokenizing word and sentences , POS (part of speech tagging) tagging , stemming, custom parsing (chunking) and stop word removal.
2. Stopwords and punctuation: Depending on problem being addressed one might remove all or some of the stop words and punctuations.
  - In our case, Agent will override NLTK's default stopwords list to retain words like 'you' or 'him' or 'herself' that can be helpful in determining context of the phrase. However, words like 'a, the, is, too' that contribute very little to the information content, will be removed.
  - Also certain punctuations, such as question marks, text equivalents for emoji's (e.g.:?, ☺, ☹, !!!), are relevant to the content, so Agent will retain those and convert it into its implied meaning (a form of metadata).
3. Stemming and Lemmatizing: (both available in NLTK) allows Agent to store only word's root, thus giving a meaningful abstraction of content being stored. Eg: Stemming will treat "apply", "applied", and "applies" as same. Though Lemmatizing (which implies a broader scope of fuzzy word matching e.g. : lemmatize(best) => good) doesn't work for verbs that well, it is useful for nouns.

## Dialogue Manager and Response generation (Q7-10)

- This module creates a unique session for each user and maintains this session until conversation ends. It keeps track of requests/responses exchanged in the session to learn the context of the conversation, avoid circular exchanges and to resolve referents.
- After preprocessing by Input-Processor module, Dialogue-Manager receives user input. It will reformat the query to resolve any referents. Eg: Agent should be able to resolve word 'it' in follow up query to 'CS7637' in example below and reformat it to '*is there any other course like CS7637*' before invoking Knowledge Engine.

<u>Resolving referent</u>	<u>Circular exchange</u>
<p>User : what do you think about <u>CS7637</u>?</p> <p>Agent : <i>CS7637 is a great course.</i></p> <p>User: is there any other course like <u>it</u>?</p> <p>Agent : <i>hmmm...</i></p>	<p>User : my GTECH email account is locked. I need help with password reset.</p> <p>Agent : <i>go to link x and follow password reset option.</i></p> <p>User: I did that but it asks me to enter my GTECH-Id, which I don't remember.</p> <p>Agent : <i>GTECH-Id was emailed two weeks back. Check your emails.</i></p> <p>User : but my account is locked. I need help with password reset.</p> <p>Agent : <i>go to link x and follow password reset option.</i></p>
	<p>User </p> <p>Lower the weight of this response with each use.</p>

- One way to deal with ‘circular exchanges’ is to lower the weight of the rule that has been fired before. Eg: In above example, weight of the rule that gives response ‘*go to link x and follow password reset option*’ should be lowered exponentially with each use. That way Agent knows not to use it when its weight becomes zero and look for next closest rule.
- It performs Topic detection and retrieves relevant script from its repository (details in section below).
- It returns the response (listed in retrieved script) to user-interaction module. If no script is found, it returns a default response and prompts user for clarification and/or on a similar topic.
  - Maintaining history of conversation enables Dialogue-Manager to learn context of the conversation and pick cues about user’s preferences and personality. This capability can help agent tweak its response in a manner that is conducive to user engagement. It can also help agent to guide the conversation in scenario where it couldn’t find an appropriate response to user’s input.
  - Detecting one’s online personality and adapting one’s response to that personality is no trivial task. Studies have shown that pronouns, articles, conjunctions and other function words are more indicative of one’s personality than content words (nouns, verbs, adjectives, and most adverbs). In English, there are close to 500 function words, and about 150 are really common. In its basic form, Agent rely on these studies to guess and adapt its response to user’s personality. But this approach is not guaranteed to be very accurate and hence warrants research and implementation of more sophisticated personality detection techniques.

## Knowledge Retrieval (Q3 & 5)

This section describes step 4 and 5 of workflow (on page-2) in more detail, especially Topic detection and Sentence Similarity measure.

- *Knowledge Engine has scripts organized by topics. Dialogue-Manager passes user query and suggested topic (related to given query) to Knowledge Engine.*
- *Knowledge Engine scans its script repository related to this topic and retrieves most relevant script. It uses Sentence Similarity measure to check similarity between given query and script in its repository.*

### **1. Sentence Similarity**

This metric is based on approach used by O’shea [2] which is a weighted aggregate of semantic similarity and word-order similarity (*semantic similarity is given higher weight*).

$$\text{Sentence similarity} = \alpha * \text{semantic similarity} + (1-\alpha) \text{ word-order similarity}$$

#### **1a. Semantic similarity:**

For this task Agent uses WordNet as a Lexical Knowledge Processor. WordNet is a database of English words that are linked together by their semantic relationships. It can be thought of a tree structure where degree of abstraction reduces at each level. It has Synsets (synonym sets) which form relations with other synsets, creating a hierarchy of concepts, ranging from very general (‘entity’) to moderately abstract (‘plant’) to very specific (“banksia-rose”) as seen in example below.

## WordNet hierarchy

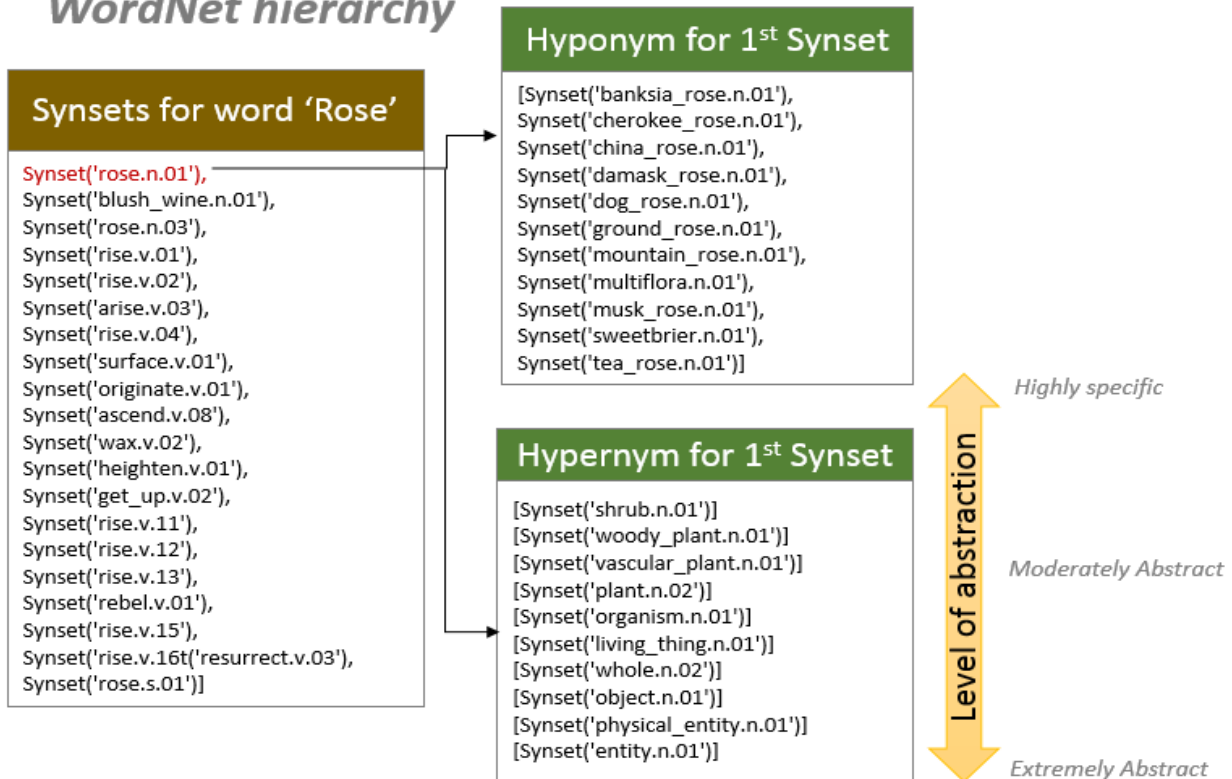


Figure 3

Given that these Synsets can be represented as graph, similarity between them can be measured based on the length of separation between them. NLTK provides several similarity measures –two of which are shown below. (*Path similarity and Wu Palmer similarity*)

Eg: This example is inspired from video lecture where Prof. Goel explained the challenge of finding meaning of a word w.r.t its context. Word 'Eat' can take multiple meanings. Eg: 1) *She ate her lunch.* 2) *Stress is eating her.* Conversely, multiple words can mean same thing.

In this example I am using WordNet to find how closely '*She ate her lunch*' and '*she ingested her lunch*' are semantically similar. It is evident from similarity score (right most columns below) that first 3 meanings of 'eat' are more closely correlated with 'ingest' when compared to last 3. (Score of 0=> no similarity, 1=> max similarity)

Synonym Set for 'ingest'	synset(x).definition()
synset('consume.v.02')	serve oneself to, or consume regularly

Synonym Set for 'eat'	synset(x).definition()
synset('eat.v.01')	take in solid food
synset('eat.v.02')	eat a meal; take a meal
synset('feed.v.06')	take in food; used of animals only
synset('eat.v.04')	worry or cause anxiety in a persistent way
synset('consume.v.05')	use up (resources or materials)
synset('corrode.v.01')	cause to deteriorate due to the action of water, air, or an acid

WordNet similarity	
w1[x]. path_similarity (w2[0])	w1[x]. wup_similarity (w2[0])
0.50	0.40
0.50	0.40
0.50	0.40
0.14	0.25
0.14	0.25
0.20	0.33

high similarity

low similarity



### 1b. Word-Order Similarity

T1 = 'She had dinner after dessert'

T2 = 'He had dessert after dinner'

Joint word set = 'She had dinner after dessert He'

# Each word is assigned a unique number based on its order of appearance in sentences

1	2	3	4	5	6
She	had	dinner	after	dessert	He

r1= {12345}      r2= {62543}

Word\_order = 1.0 - (numpy.linalg.norm (r1 - r2) / numpy.linalg.norm (r1 + r2))

### 2. Topic/Track detection:

Dialogue-Manager is boot-strapped with relevant topic names such as admission, registration, course info, tuition, transfer credit etc. and relevant words for each topic. This list will grow as Agent is trained on human chat history and/or customer-service ticketing database.

Eg :      { 'Admission'      : 'credentials', 'admission criteria', 'GRE', 'TOEFL', 'transcripts', 'Citizenship' }  
         { 'Program-Info'    : 'curriculum', 'academic calendar', 'specialization' }  
         { 'tuition'          : 'fee', 'financial aid', 'student-loan', 'program cost' }

Topic detection is performed by Dialogue-Manager by extracting keywords from user input and pattern matching it with this preconfigured list.

This approach can be refined by leveraging abstraction through the WordNet Hypernym Hierarchy. It consists of lifting the words extracted from user input to a more abstract equivalent by traversing the tree in upward direction.

Eg: using Fig-3, if Agent sees a word 'Rose' in input query, it can abstract it via checking its hypernyms hierarchy which tells that Rose is-a shrub ->woody plant->vascular plant->organism->living thing ... and so on. This will help Agent to approximate theme of the conversation.

## Wrap up

This paper presented high level design of an agent that tries to engage in a conversation akin to human, by learning context of the conversation and adapting its response to 'connect' with the user. This design makes some sweeping assumptions with regards to knowledge acquisition, discourse coherence and user's personality detection, which clearly needs to be enhanced in future.

## References

1. *Artificial Intelligence* by Patrick Henry Winston
2. *A Novel Approach for Constructing Conversational Agents using Sentence Similarity Measures*
3. <http://wordnet.princeton.edu/>
4. <http://www.nltk.org/howto/wordnet.html>
5. CS7637 Udacity Video lectures