Overview of Chatbots – Assignments in CS 410 – Text Information System

Sudipto Sarkar
The Grainger College of Engineering, Department of Computer Science, University of Illinois – Urbana Champaign, 201 North Goodwin Avenue, Urbana, Illinois 61801, USA,
Technology Review – CS 410 Text Information Systems
sudipto2@illinois.edu

***Abstract*: The objective of this document is to provide a high-level understanding of chatbots. Key topics covered in this document are introduction to Chatbot, its brief history, the general architecture of chatbot, different types of Chatbots, their development approaches and finally some of the challenges we face with the current Chatbots. Since this course is tied to Information Retrieval, we will only be looking at the implementation of a simple Rule based chatbot system which uses the core Information Retrieval techniques to look for an optimal response to a user query.**

I. Introduction

Chatbot is a computer application that is capable of a two-way conversation with a user in form of text or speech. Behind the scenes it uses Natural Language processing and sentiment analysis to answer user queries. Chatbots can be found in wide range of industries like education, business, e-commerce, health, and entertainment. They serve many purpose ranging from customer support, helping in therapy to being a day-to-day digital assistant for the user. The primary motivation for chatbot is to improve user productivity by reducing service costs and handling multiple requests simultaneously.

II. Brief History of Chatbot

[2] In 1950, Alan Turing envisioned what it would be like if a computer program could interact with humans. He developed the Turing Test which is the first idea of a chatbot. The first actual chatbot was built in 1966 and was named ELIZA. Its ability was limited as it used simple pattern matching and a pre-defined template-based response. Later there were many other improved chatbots developed namely PARRY (1972), ALICE (1995) which relied on AIML, SmarterChild (2001) and the most modern ones that we currently use – Apple Siri, Microsoft Cortana, Amazon Alexa, Google Assistant, and IBM Watson.

III. Chatbot General Architecture

A chatbot needs to perform the below key basic tasks:

- Parse the user input
- Interpret what the user input means
- Provide an appropriate response or output

[1][2][3]A good chatbot will be able to identify the intent and entities of the query. The intent is the purpose or category of the user query. Entities are extra information that describes the user's intent. With these pieces of information, chatbots should be able to respond to the user with a response.

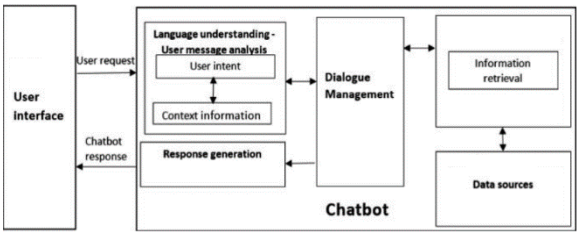
Below is the flow of chatbot system from user query request to chatbot response:

- User sends in the request from front end
- The chatbot receives the user request, the Language Understanding Component parses it to infer the user's intention
- Once chatbot analyzes the query and reaches the best interpretation it can, it will perform one of below:
    - Respond to user with an answer
    - Remember what it parsed/understood and wait for next user interaction to happen
    - Respond to user with further clarification to understand the context better
- If the chatbot can understand the request, then information retrieval based on the query takes place. Chatbot gathers response data from different data sources.
- The Response Generation Component of chatbot then prepares to generate the response with the data it has gathered using Natural Language Generation. Responses are provided using any of the three models – Rule Based, Retrieval Based or Generative Model.

Overview of Chatbots – Assignments in CS 410 – Text Information System

Sudipto Sarkar
The Grainger College of Engineering, Department of Computer Science, University of Illinois – Urbana Champaign, 201 North Goodwin Avenue, Urbana, Illinois 61801, USA,
Technology Review – CS 410 Text Information Systems
sudipto2@illinois.edu

- Dialogue Management Component - It controls and updates the conversation context. It keeps the current intent and the identified entities until that point of the conversation. If the chatbot is unable to collect the necessary context information, it asks for additional context information from the user to fulfill missing entities. It also asks follow-up questions after the intent is recognized.



IV. Chatbot Development Approaches

[2][3]Chatbots can be developed using either of two approaches – Pattern Matching and Machine Learning Approaches.

- Pattern Matching Approach: In this mode the chatbots match the user input to a rule pattern and select the response from a predefined set based on the pattern matching algorithm. In this approach, a query will always have a fixed response for a specific query. Hence the more extensive the rule collection is, the more capable the system would be to answer various queries. Downside of this approach is that the system does not have the natural human interaction touch to it. However, the constraint on the rule and response means faster and quality responses. Some of the most common languages used to build a Pattern based chatbot includes Artificial Intelligence Markup Language (AIML), RiveScript and ChatScript[2].
    - AIML follows a stimulus-response approach. It is an XML-based markup language and is tag-based. Basic units of dialogues in AIML are called Categories (Tag <category>) which are formed by user input patterns (tag <pattern>) and chatbot responses (tag <template>). AIML is sometimes used in conjunction with Latest Semantic Analysis (LSA). AIML may answer questions based on specific templates and for questions that cannot be answered this way, LSA is used for responses.
    - RiveScript is a plain text, line-based scripting language for the development of chatbots and other conversational entities. It is open source with available interfaces for Go, Java, JavaScript, Perl, and Python.
    - ChatScript is a successor to AIML and is a system that consists of an open-source scripting language and engine that runs it. It is comprised of rules which are associated with topics, finding the best item that matches the user query string and executing a rule in that topic. It matches user inputs to chatbot outputs using pattern matching. ChatScript uses concepts that are collections of similar words concerning the meaning and other parts of speech. ChatScript also includes long term memory that stores user information that can be leveraged during query response.
- Machine Learning Approaches: In this approach the Chatbot extracts content from user query using Natural Language Processing (NLP). It considers the whole dialog context and not just the current query and does not require pre-defined set of rules or responses unlike Pattern based approach. However, they need extensive dataset to be trained with which is one of the key challenges in terms of availability.
    - Natural Language Processing (NLP) – It is an area of Artificial Intelligence where knowledge of the understanding of human language is gathered to

Overview of Chatbots – Assignments in CS 410 – Text Information System

Sudipto Sarkar
The Grainger College of Engineering, Department of Computer Science, University of Illinois – Urbana Champaign, 201 North Goodwin Avenue, Urbana, Illinois 61801, USA,
Technology Review – CS 410 Text Information Systems
sudipto2@illinois.edu

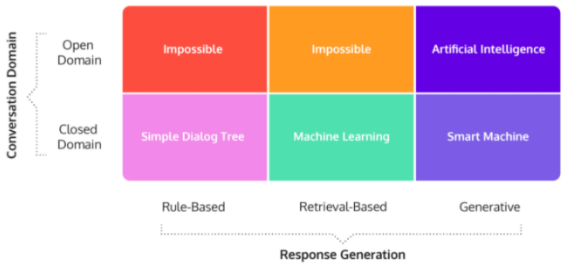
develop techniques that will enable computers to understand human interaction and respond to user queries.

- Natural Language Understanding (NLU) – It is used to retrieve context from the unstructured user input in human language and respond based on the current user's intention. NLU supports intent classification and entity extraction, considering the context information.
- Artificial Neural Networks - Both Retrieval and Generative-based models use different kinds of Artificial Neural Networks. The system takes the user's input, computes its vector representations, feeds it as features to the neural network, and produces the response. In this system words are mapped into vector representations and this is called word embedding. Word2Vec is one of the predominant techniques used in deep learning for word embedding. In Retrieval-based systems, Artificial Neural networks, takes the user's input vector and give a probability of every intent. The classification of entities in the user input is done by Named-Entity Recognition (NER) systems.
- Recurrent Neural Networks – In this model the previous context of the conversation is considered. User input and information from prior chats are fed to the network. The model uses Long Short Term Memory networks (LSTMs) dependencies where the look behind on information can be configured via parameters.
- Sequence to Sequence Model - Typical example of a Generative model is Sequence-to-Sequence which generates a target sequence by looking at the source sequence. The source sequence is the user's input, and the target sequence is the chatbot's response.
- Deep Sequence to Sequence (Seq2seq) models provide better human like performance and are also an example of a Generative model where it has more parameters.

V. Chatbot Categories

[1] Chatbots can be categorized based on the purpose they are designed to serve. The purposes can vary from Goals it serves, Knowledge domain it pertains to, Service it provides, Response generation methods, Permission level of the bot and Mode of Communication of the chatbot.

Below are some key categories to investigate [1].



- Response based: This first category is based on how the chatbot generates the response to the user query. This is further sub-divided into Rule based, Retrieval-Based and Generative approaches.
  - Rule based approach is the simplest of chatbots and the responses are pre-defined and are generated based on a series of rules. Decision trees are used in this approach and there are a clear set of outputs defined for each step of the query. Below is a sample implementation of a simple Rule based Chatbot with detailed steps [4]:
    - Create a corpus: The below code snippet will use the

Overview of Chatbots – Assignments in CS 410 – Text Information System

Sudipto Sarkar
The Grainger College of Engineering, Department of Computer Science, University of Illinois – Urbana Champaign, 201 North Goodwin Avenue, Urbana, Illinois 61801, USA,
Technology Review – CS 410 Text Information Systems
sudipto2@illinois.edu

below URL as the source to create the corpus. It extracts all the paragraphs from the article and convert to lower case for easier processing.

```
# Create corpus
raw_html = urllib.request.urlopen('https://en.wikipedia.org/wiki/Chatbot')
raw_html = raw_html.read()

article_html = bs.BeautifulSoup(raw_html, 'lxml')
article_paragraphs = article_html.find_all('p')
article_text = ''

for para in article_paragraphs:
    article_text += para.text

article_text = article_text.lower()
```

- Text Preprocessing: This step preprocesses the above corpus text to remove special characters, empty spaces. It also divides the sentences and words into different collections. It finally removes the punctuations and lemmatizes the text

```
# Text Preprocessing and Helper Function
article_text = re.sub(r'\[[0-9]*\]', ' ', article_text)
article_text = re.sub(r'\s+', ' ', article_text)

article_sentences = nltk.sent_tokenize(article_text)
article_words = nltk.word_tokenize(article_text)

# Lemmatize
wnlemmatizer = nltk.stem.WordNetLemmatizer()

def perform_lemmatization(tokens):
    return [wnlemmatizer.lemmatize(token) for token in tokens]

punctuation_removal = dict((ord(punctuation), None) for punctuation in string.punctuation)

def get_processed_text(document):
    return perform_lemmatization(nltk.word_tokenize(document.lower().translate(punctuation_
```

- Respond to user greetings: To differentiate greetings from normal queries, simpler greeting responses are created that will match up a pre-defined set of words and respond with a pre-defined reply. A dedicated Greetings Method has been created to handle these.

```
# Responding to Greetings

greeting_inputs = ("hey", "good morning", "good evening", "morning", "evening", "hi", "whatsup")
greeting_responses = ["hey", "hey hows you?", "*nods*", "hello, how you doing", "hello", "Welcome, I am gooc

def generate_greeting_response(greeting):
    for token in greeting.split():
        if token.lower() in greeting_inputs:
            return random.choice(greeting_responses)
```

- Respond to User Queries: If the queries are anything other than greetings, the response is generated based on the cosine similarity of the vectorized form of the input sentence and the sentences in the corpus. TfidfVectorizer and cosine_similarity functions are used for this. The method below takes the user input and finds the cosine similarities of the user query with sentences in the corpus. The user input is appended to the end of the list of existing sentences. “word_vectorizer” contains the word vector for the user input (corpus sentences and user entered sentence). “tfidfvectorizer” converts all the sentences in the corpus along with the input sentence into their corresponding vectorized form. “similar_vector_values” contains the cosine similarity between the last items in all_word_vectors, which is the user query, with the rest of the corpus sentences. “similar_sentence_number” sorts the cosine similarity vector output in such a way that the second last item is the item that matches the user query (the last item) the most. If the similarity match results in “0” that means, there was no match. Else if the result is nonzero, then then second last sentence in the vector is the answer to the user’s query and is returned.

Overview of Chatbots – Assignments in CS 410 – Text Information System

Sudipto Sarkar
The Grainger College of Engineering, Department of Computer Science, University of Illinois – Urbana Champaign, 201 North Goodwin Avenue, Urbana, Illinois 61801, USA,
Technology Review – CS 410 Text Information Systems
sudipto2@illinois.edu

```
# Respond to user queries
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def generate_response(user_input):
    chat_response = ''
    article_sentences.append(user_input)

    word_vectorizer = TfidfVectorizer(tokenizer=get_processed_text, stop_words='english')
    all_word_vectors = word_vectorizer.fit_transform(article_sentences)
    similar_vector_values = cosine_similarity(all_word_vectors[-1], all_word_vectors)
    similar_sentence_number = similar_vector_values.argsort()[0][-2]

    matched_vector = similar_vector_values.flatten()
    matched_vector.sort()
    vector_matched = matched_vector[-2]

    if vector_matched == 0:
        chat_response = chat_response + "I am sorry, I could not understand you"
        return chat_response
    else:
        chat_response = chat_response + article_sentences[similar_sentence_number]
        return chat_response

word_vectorizer = TfidfVectorizer(tokenizer=get_processed_text, stop_words='english')
all_word_vectors = word_vectorizer.fit_transform(article_sentences)

similar_vector_values = cosine_similarity(all_word_vectors[-1], all_word_vectors)
similar_vector_values = cosine_similarity(all_word_vectors[-1], all_word_vectors)
```

- Chatting with the bot: Here an interactive channel is opened to chat with the bot. The function will continue looping and keep asking for user input via the input() function till the user types in "bye/thanks/thank you very much/thank you" to exit the system.

```
# Chatting
continue_dialogue = True
print("Hello, I am your friend AskMeBot. You can ask me any question regarding Chatbo
while(continue_dialogue == True):
    human_text = input()
    human_text = human_text.lower()
    if human_text != 'bye':
        if human_text == 'thanks' or human_text == 'thank you very much' or human_tex
            continue_dialogue = False
            print("AskMeBot: Most welcome")
        else:
            if generate_greeting_response(human_text) != None:
                print("AskMeBot: " + generate_greeting_response(human_text))
            else:
                print("AskMeBot: ", end="")
                print(generate_response(human_text))
                article_sentences.remove(human_text)
    else:
        continue_dialogue = False
        print("AskMeBot: Good bye and take care of yourself...")
```

  - In Retrieval-Based approach, the responses are also pulled from an existing corpus. Machine Learning models like Statistical NLP models are used to interpret the user query and generate the most fitting response. Although this approach still relies on a pre-defined set of responses, the advantage this approach has over Rule based is that it can self-learn and improve over time.
  - Generative based – In this approach the chatbots can generate their own response based on user input in place of looking up a pre-defined set of responses. Hence, they pose a significant challenge in implementation as they rely on getting trained using large volumes of data. It is often not clear what rules are used for their decision making which makes them prone to errors. In contrast the Retrieval based models guarantee quality of responses since as they are pre-defined. Chatbots sometimes used a mixture of Generative and Retrieval based approaches to produce optimal results. Customer service chatbots might use Generative approach to have a more open-ended conversation with the end user for general topics but use the Retrieval based approach to answer specific question directed towards a product.

- Conversation Domain based: Chatbots can also be categorized based on the range of conversation topics.
  - Closed Domain chatbots focus on certain specific topics. They are designed with a certain topic in mind and hence are efficient and product quality output as their domain of response is limited. Example would be chatbots for mobile phone companies, travel site chatbots etc.
  - Open Domain chatbots or conversational agents can initiate and explore any range of topics just like a natural human interaction. Hence, they are often used in areas of therapy to have an emotional connection with the user. This also poses challenges in evaluating and implementing them.
- Initiative based:
  - Chatbots are also categorized based on which party – user or system, initiated

Overview of Chatbots – Assignments in CS 410 – Text Information System

Sudipto Sarkar
The Grainger College of Engineering, Department of Computer Science, University of Illinois – Urbana Champaign, 201 North Goodwin Avenue, Urbana, Illinois 61801, USA,
Technology Review – CS 410 Text Information Systems
sudipto2@illinois.edu

the conversation. A Hotel reservation system is an example of a "Mixed Initiative" chatbot where the user is freely able to ask any question and the system can analyze the query and answer appropriately. Hence there is equal participation from both ends in taking the initiative. It represents a more human like conversation where either party take the initiative.

- In contrary a System Initiative chatbot is the one where the application controls the discussion by explicitly asking the questions. Here the user responses are anticipated as the question topics are targeted. However, the system lacks flexibility and naturalness of a human interaction.

VI. Limitations of Chatbots

[2][3] Even though bots provide significant advantages, they have their own set of drawbacks:

- Failure in intent understanding – Chatbots often fail to understand the intent of the user. This can drive a customer away if this is a sales related assistant.
- Toxic content in user input – Recording personal data by unreliable services the chatbot is integrated to classifies as toxic content. Copyright theft is another example of toxic content on Chatbots. Chatbots should be trained to avoid misuse of toxic content.
- Deception towards chatbot – Detecting deceptions is crucial in Chatbot applications. Human deceivers might hide their deception while interacting with a Chatbot. For example, the Alexa service relies only on weak single-factor authentication, which can be broken because it follows voice commands without access control dependent on physical presence.

VII. Ethics in Chatbot

[1] There are still certain ethical issues that arises and below are the key ones to investigate:

- User Transparency: User needs to be aware of all aspects involving the chatbot and consequences of interacting with one. A common concern is the privacy and protection of user data. Depending on what regulations are put in place, any information that the users share with the bot during their conversation could potentially be collected, used, or sold without their consent. Thus, it is key to be open and clear about data usage, ownership, and protection.
- Chatbot Persona: One of the biggest controversies surrounding Chatbots is the assignment of gender. Chatbots are disproportionately given female names or voices. Chatbot developers need to be careful to avoid gender bias during the design of the bot. Additionally we need to take caution when training the bot to ensure that it behaves appropriately. If it is not properly trained, the chatbot could be at risk of displaying racism, sexism, or use of abusive language.
- Art of Communication: For a Chatbot there is more to communication than simply providing a response. Users communicating with Chatbots in a negative or abusive way is not uncommon. Accepting the abuse may encourage user behavior. Hence Chatbot developers needs to design bots that can actively tackle harassment and turn the situation around. Similarly, some users might reach out to Chatbots as a source of company or comfort and these users may grow attached and Chatbots should be particularly considerate and empathetic towards the user.

VIII. Concluding Remarks

Chatbots help in minimizing the human interference in most of our daily routines and at the same time are great information gathering tools. Thereby they

Overview of Chatbots – Assignments in CS 410 – Text Information System

Sudipto Sarkar
The Grainger College of Engineering, Department of Computer Science, University of Illinois – Urbana Champaign, 201 North Goodwin Avenue, Urbana, Illinois 61801, USA,
Technology Review – CS 410 Text Information Systems
sudipto2@illinois.edu

provide significant savings in operations management. In recent years there has been significant improvements with integration to Machine Learning. This will eventually lead to more jobs getting automated. Chatbots hold great potential and it is an exciting area to be interested in!

References:

1) https://www.codecademy.com/learn/paths/build-chatbots-with-python
2) https://www.sciencedirect.com/science/article/pii/S2666827020300062
3) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7256567/
4) https://stackabuse.com/python-for-nlp-creating-a-rule-based-chatbot/