VYTAUTO DIDŽIOJO
UNIVERSITETAS
MCMXXII

CHATBOTS & VIRTUAL ASSISTANT'S SETUP DEVTOOLS & EXAMPLES

BY: HARSHAL SANTOSH SONAWANE

PROFESSOR: VYTAUTAS BARZDAI

SUBJECT: INTERNET TECHNOLOGY

TERM PAPER [IF160097]

VYTAUTAS MAGNUS UNIVERSITY

DEPARTMENT OF INFORMATICS

Contents

1 INTRODUCTION TO CHATBOTS

Chatbots are online human-computer dialog systems with natural language. The first conceptualization of the chatbot is attributed to Alan Turing, who asked “Can machines think?” in 1950. Since Turing, chatbot technology has improved with advances in natural language processing and machine learning. Likewise, chatbot adoption has also increased, especially with the launch of chatbot platforms by Facebook, Kik, Slack, Skype, WeChat, Line, and Telegram. By September 2016, Facebook Messenger hosted 30,000 bots and had 34,000 developers on its platform. The Kik Bot Shop announced in August 2016 that the 20,000 bots created on its platform had “exchanged over 1.8 million messages.”

1.1 USAGE OF CHATBOTS

“Its usage is potentially endless” many sectors like E-commerce, Finances, HR, Healthcare each and every place where there is a need for certain information to be responded back or certain action to be taken based on simple text or voice query by the user. Also, Chatbots Market worth 3,172.0 Million USD by 2021 which will only increase based on the NLP improvements and the AI improvements. Also, Chatbots expected to cut business costs by $8 billion by 2022.

1.2 TYPES OF CHATBOTS

There are two types of Chatbots

1) Rule-Based Chatbots
2) AI- Based Chatbots

- **Rule-Based Chatbots**: A rule-based chatbot is provided with a list of set answers for a set of queries a user might use the chatbot for. For example, if a user asks question X, the chatbot will answer Y. But the problem arises when the chatbot is having a query asked in a different way. An example would be
  Q1: How is the weather today?
  Q2: Is it hot today?
  Q3: Is it cold today?
  Q4: Is it a good time to go out for a walk today?

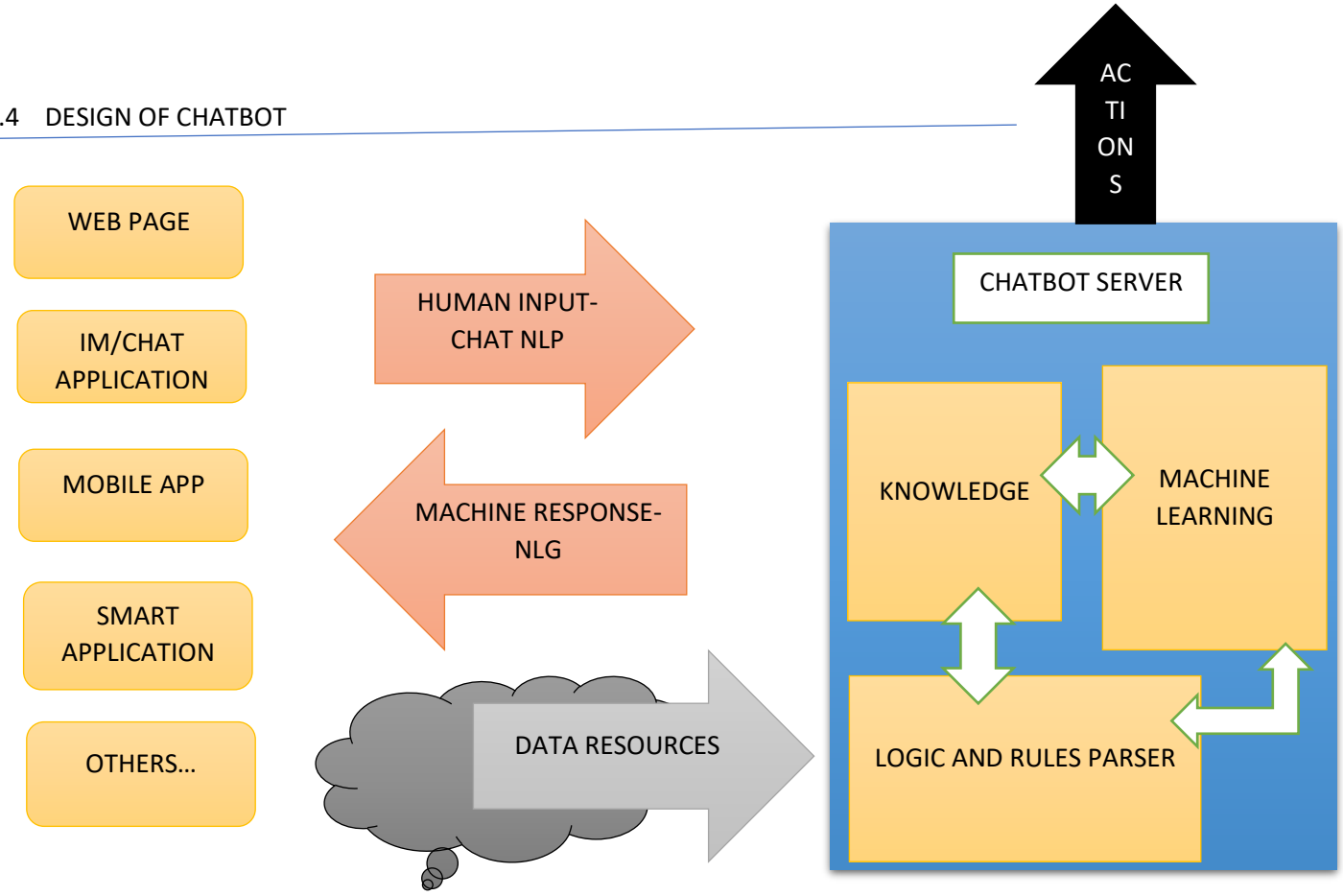
- All four of these are related to the weather but the problem is that we need to make sure that all four of them to be entered there to give the same response. Imagine 100 similar ways of asking the same question then we need to enter all 100 types of question for a single response.
- Here, in this case, the NLP or NLU implementation is not present. If a question is asked in a new way where it has not been programmed then it will point to default intent where it will display mostly as “Sorry I didn’t understand it” which will affect user experience.

- **AI-Based Chatbots:** These bots learn independently on their own based on the data provided and keep on learning and improving based on previous interactions. In this method, chatbots will learn to interact properly over a period of time. It needs to be interactive as it needs to respond like a human over a period of time. The data that is fed or the responses were given needs to be controlled otherwise it will lead to chatbots developing a negative personality.

1.3 20 Best platforms to build Chatbots

1. IBM Watson Conversation Service
2. AgentBot
3. Twyla
4. Pypestream
5. Live Agent
6. Digital Genius
7. Semantic Machines
8. Msg.ai
9. Wit.ai
10. Rasa NLU
11. Api.ai
12. Microsoft Bot framework
13. MLUIS
14. Chatfuel
15. PandoraBots
16. Chatterbot
17. Octane.ai
18. Rebot.me
19. Manychat
20. Flowxo

1.4 DESIGN OF CHATBOT



2 SETUP

2.1 NLTK: A Brief Intro

NLTK (Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

NLTK has been called “a wonderful tool for teaching and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

2.2 Natural Language Processing with Python: provides a practical introduction to programming for language processing. I highly recommend this book to people beginning in NLP with Python.

2.3 Downloading and Installing NLTK

1) Install NLTK: run pip install nltk
2) Test installation: run python then type import

Link for installation https://www.nltk.org/install.html

2.4 Installing NLTK Packages

Import NLTK and run nltk. download (). This will open the NLTK downloader from where you can choose the corpora and models to download. You can also download all packages at once.

2.5 Text preprocessing with NLTK:

In the NLP project we need to pre-process it to make it ideal for working. Basic text pre- processing includes:

- Converting the entire text into uppercase or lowercase, so that the algorithm does not treat the same words in different cases as different
- Tokenization: Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e words that we actually want. Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.

2.6 Bag of Words:

After the initial preprocessing phase, we need to transform text into a meaningful vector (or array) of numbers. The bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

•A vocabulary of known words.

•A measure of the presence of known words.

Why is it is called a “bag” of words? That is because any information about the order or structure of words in the document is discarded and the model is only concerned with whether the known words occur in the document, not where they occur in the document. The intuition behind the Bag of Words is that documents are similar if they have similar content. Also, we can learn something about the meaning of the document from its content alone.

For example, if our dictionary contains the words {Learning, is, the, not, great}, and we want to vectorize the text “Learning is great”, we would have the following vector: (1, 1, 0, 0, 1).

3 CHATBOT CREATION

Now Let’s start the creation of our chatbot. I will name my chatbot here as “ROBO”

3.1 IMPORTING THE NECESSARY LIBRARIES

```
import nltk
import numpy as np
import random
import string # to process standard python strings
```

3.2 Corpus

For our example, we will be using the Wikipedia page for chatbots as our corpus. Copy the contents from the page and place it in a text file named ‘chatbot.txt’. However, you can use any corpus of your choice.

3.3 Reading in the data

We will read in the corpus.txt file and convert the entire corpus into a list of sentences and a list of words for further pre-processing.

```
f=open('chatbot.txt','r',errors = 'ignore')

raw=f.read()

raw=raw.lower()# converts to lowercase

nltk.download('punkt') # first-time use only
nltk.download('wordnet') # first-time use only

sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences
word_tokens = nltk.word_tokenize(raw)# converts to list of words
```

example of the sent_tokens and the word_tokens

```
sent_tokens[:2]
['a chatbot (also known as a talkbot, chatterbot, bot, im bot,
interactive agent, or artificial conversational entity) is a
computer program or an artificial intelligence which conducts a
conversation via auditory or textual methods.',
 'such programs are often designed to convincingly simulate how a
human would behave as a conversational partner, thereby passing the
turing test.']

word_tokens[:2]
['a', 'chatbot', '(', 'also', 'known']
```

3.4 Pre-processing the raw text

We shall now define a function called LemTokens which will take as input the tokens and return normalized tokens.

```
lemmer = nltk.stem.WordNetLemmatizer()
#WordNet is a semantically-oriented dictionary of English included
in NLTK.

def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in
string.punctuation)
def LemNormalize(text):
    return
LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dic
t)))
```

3.5 Keyword matching

Next, we shall define a function for a greeting by the bot i.e if a user's input is a greeting, the bot shall return a greeting response. ELIZA uses a simple keyword matching for greetings. We will utilize the same concept here.

```
GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's
up","hey",)

GREETING_RESPONSES = ["hi", "hey", "*nods*", "hi there", "hello", "I
am glad! You are talking to me"]

def greeting(sentence):

    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)
```

3. 6 Generating Response

To generate a response from our bot for input questions, the concept of document similarity will be used. So we begin by importing necessary modules. From scikit learn library, import the TFidf vectorizer to convert a collection of raw documents to a matrix of TF-IDF features.

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

Also, import cosine similarity module from scikit learn library

```
from sklearn.metrics.pairwise import cosine_similarity
```

This will be used to find the similarity between words entered by the user and the words in the corpus. This is the simplest possible implementation of a chatbot. We define a function response which searches the user's utterance for one or more known keywords and returns one of several possible responses. If it doesn't find the input matching any of the keywords, it returns a response:" I am sorry! I don't understand you"

```
def response(user_response):
    robo_response=''
    sent_tokens.append(user_response)

    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize,
stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]

    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand
you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens[idx]
        return robo_response
```

Finally, we will feed the lines that we want our bot to say while starting and ending a conversation depending upon user's input.

```
flag=True
print("ROBO: My name is Robo. I will answer your queries about
Chatbots. If you want to exit, type Bye!")

while(flag==True):
    user_response = input()
    user_response=user_response.lower()
    if(user_response!='bye'):
        if(user_response=='thanks' or user_response=='thank you' ):
            flag=False
            print("ROBO: You are welcome..")
        else:
            if(greeting(user_response)!=None):
                print("ROBO: "+greeting(user_response))
            else:
                print("ROBO: ",end="")
                print(response(user_response))
                sent_tokens.remove(user_response)
    else:
        flag=False
        print("ROBO: Bye! take care..")
```

4 ENTIRE CODE:

```
import nltk

import warnings

warnings.filterwarnings("ignore")


# nltk.download() # for downloading packages


import numpy as np

import random

import string # to process standard python strings
```

```
f=open('chatbot.txt','r',errors = 'ignore')
raw=f.read()
raw=raw.lower()# converts to lowercase
#nltk.download('punkt') # first-time use only
#nltk.download('wordnet') # first-time use only
sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences
word_tokens = nltk.word_tokenize(raw)# converts to list of words


sent_tokens[:2]


word_tokens[:5]


lemmer = nltk.stem.WordNetLemmatizer()
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))


GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up","hey",)
GREETING_RESPONSES = ["hi", "hey", "*nods*", "hi there", "hello", "I am glad! You are talking to me"]
```

```
# Checking for greetings
def greeting(sentence):
    """If user's input is a greeting, return a greeting response"""
    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)



from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity



# Generating response
def response(user_response):
    robo_response=''
    sent_tokens.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens[idx]
```

```
        return robo_response



flag=True

print("ROBO: My name is Robo. I will answer your queries about Chatbots. If you want to exit, type Bye!")


while(flag==True):
    user_response = input()
    user_response=user_response.lower()
    if(user_response!='bye'):
        if(user_response=='thanks' or user_response=='thank you' ):
            flag=False
            print("ROBO: You are welcome..")
        else:
            if(greeting(user_response)!=None):
                print("ROBO: "+greeting(user_response))
            else:
                print("ROBO: ",end="")
                print(response(user_response))
                sent_tokens.remove(user_response)
    else:
        flag=False
        print("ROBO: Bye! take care..")
```

4.1 Now, Let us see how it interacts with humans:

```
e the chatbot personality, the questions that will be asked to the users, and the overall interaction.it can be viewed as a sub
set of the conversational design.
Describe chatbot design?
ROBO: design
the chatbot design is the process that defines the interaction between the user and the chatbot.the chatbot designer will defin
e the chatbot personality, the questions that will be asked to the users, and the overall interaction.it can be viewed as a sub
set of the conversational design.
```

5 REFERENCES

1) https://www.ijcseonline.org/pub_paper/27-IJCSE-02149.pdf
2) https://www.chatbot.com/docs
3) https://www.chatbot.com/docs
4) https://chatbotslife.com/so-you-want-to-build-a-chatbot-from-scratch-part-1-5bec5cb866b0
5) https://en.wikipedia.org/wiki/Chatbot