Chatbot Technical Docs

Code

Purpose: To build a chatbot in telegram according to emergency evaluation methods used by doctors.

Code structure The chatbot set consist of a set of libraries that supports the main telegram bot code in the file chatbot_telegram.

Function of supporting libraries: NewConditionQuestions: To produce a sequence of questions provided by doctor in accordance to emergency evaluation methods. The recording functions that access stored data and then record them onto PDF and TXT files are also found here. It is supported by the PDF libraries found in reingart-pyfpdf-cfe3d03 to help document the conversation into PDF file. The main libraries used for the chatbot's recording function are moved out to the Final Chatbot set file. These files consist of: *init* Fonts Fpdf Html Php Py3k Template ttfonts

RasaNLU: The NLP capabilities of the chatbot comes from this library. It is used to train the code and also to load the learnt model into the chatbot. Any new models trained will appear in the projects folder under default. The other 2 rasafiles are rasa libraries called into RasaNLU file to train and load the models. The training file must be in json format which can be easily made using https://yuukanoo.github.io/tracy/#/agents (https://yuukanoo.github.io/tracy/#/agents). The chatbot current training file is demo-rasa1 file found in Final Chatbot set.

Workings of Code Chatbot_telegram:

To download telegram bot api go to https://github.com/python-telegram-bot/python-telegram-bot (https://github.com/python-telegram-bot/python-telegram-bot) and read ReadMe

Recording Telegram API stores information in a conversation through python dictionary user_data. Information of conversation are stored using user_data[“relevant” name]= text. At the end of a conversation, the recording function is called in to produce the content in a pdf file.

Hide

```
def EndOfCircumcision(bot,update,user_data):
    reply_keyboard = [['Yes','No']]
    reply_keyboardH=[[ "Helpful", "Not Helpful"]]
    user = update.message.from_user
    text = update.message.text
    user_data["painmed"]=text
    if user_data["bleed"]=="Bleeding" and text=="No":
        update.message.reply_text("Please come to our Childrens' Emergency at 5 Lower Kent Ridge
 Road, 119074. One of our Specialist Nurse will be calling your registered Phone Number to check
 on you in 10 mins.")
        if user_data["helpfulcount"]==2:
            number=user_data["number"]
            cd.recordred2(number,user_data)
        else:
            cd.recordend(user_data)
        return ConversationHandler.END #### CONVERSATION ENDING Record function called in the st
ep above
    else:
        update.message.reply_text("Educational resource on wound dressing")
        time.sleep(10)
        update.message.reply_text("Was this helpful?",
                            reply_markup=ReplyKeyboardMarkup(reply_keyboardH, one_time_keyboard=
True))
    return ENDOFCIRCUMCISION
```

Keeping track of conversation conditions The user_data dictionary is also used to keep track of the state of the conversation, to bring over relevant information over to the next state of the conversation. E.g user_data[helpfulcount] under helplful is used to keep track on whether the user is going through the conversation the first or second time, asking relevant questions again the first time or ending the conversation the second time. Everytime when the conversation starts again, underCheckReocurrence function, if the receiving text is tryagain, the chatbot will save important data, clear user_data dictionary and readd the important data back to user_data dictionary. This is important as when a user wants to try again, it is possible for existing data in the dictionary from the previous attempt to affect the conditioning of the question flow.

Hide

```
def CheckReoccurence(bot, update, user_data):

    user = update.message.from_user
    text = update.message.text

    if text=="Yes" or text=="No":
        user_data["similar?"]=text
        previous=user_data["Past Symptoms"]

    if text=="Yes":
        update.message.reply_text('A reoccurence of '+previous+" can have serious implications.
 Please come to our Childrens' Emergency at 5 Lower Kent Ridge Road, 119074. One of our Speciali
st Nurse will be calling your registered Phone Number to check on you in 10 mins.")
        if user_data["helpfulcount"]==2:
            number=user_data["number"]
            cd.recordred2(number,user_data)
        else:
            cd.recordend(user_data)
        return ConversationHandler.END
    elif text=="Stop":
        update.message.reply_text("Okay, please bring your child to A&E asap for assistance!")
        cd.recordnotagain(user_data)
        return ConversationHandler.END
    elif text=="No":
        update.message.reply_text("Alright, What seems to be the problem")
    else: ###where i want to try again
        number=random.randrange(0,10000000) ###generate random patient number
        cd.recordtryagain(number,user_data) ####Record first attempt
        age=user_data["age"]
        ts=time.time()
        starttime=datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
        Surgery=user_data['Surgery?'] ####Storing data from dictionary before clearing them
        if Surgery=="Yes": ###conditioning on question flow so that non existing dictionary keys
 will not be called
            surgery=user_data['surgery']
            score=user_data["score"]
            if surgery=="Circumcision":
                bleed=user_data["bleed"]
                puswound=user_data["puswound"]
                painmed=user_data["painmed"]
            if surgery=="Biliary Drainage":
                    drain=user_data["drain"]
                    string=user_data["string"]
        if Surgery=="No": ###conditioning on question flow so that non existing dictionary keys
 will not be called
            hospital=user_data["hospital?"]
            if hospital=="Yes":
                pastsymptoms=user_data["Past Symptoms"]
                if pastsymptoms!="None":
                    similar=user_data["similar?"]
        user_data.clear() #### Clear dictionary and readding them with same conditions
        user_data["age"]=age
        user_data["attempt"]=0
        user_data["starttime"]=starttime
        user_data["Surgery?"]=Surgery
        if Surgery=="Yes":
            user_data["surgery"]=surgery
            if user_data["surgery"]=="Circumcision":
                user_data["score"]=score
                user_data["bleed"]=bleed
                user_data["puswound"]=puswound
                user_data["painmed"]=painmed
            elif user_data["surgery"]=="Biliary Drainage":
                user_data["score"]=score
                user_data["drain"]=drain
                user_data["string"]=string
        if Surgery=="No":
            user_data["hospital?"]=hospital
            if hospital=="Yes":
                user_data["Past Symptoms"]=pastsymptoms
                if pastsymptoms!="None":
                    user_data["similar?"]=similar
        user_data["number"]=number
        user_data["helpfulness"]=1
        user_data["helpfulcount"]=1
        update.message.reply_text("Alright, What seems to be the problem")
    return CHECKREOCCURENCE
```

Redflag questions Redflag questions are questions where if answered correctly, will indicate that a redflag situation is present. These situations might have age and previous conditions present for the redflag situation to arise. Conditions are remembered in user_data and conditioned using a list where these conditions are stored. E.g under function ProcessConditions1, for gastroenteritis question, for number of times a patient vomit questions if the patient age is under 6months, and vomit/diarrhea more than 3 times in a day, it is an emergency situation. These conditions are captured in the list redflagage and redflaganswers Where they are conditioned on.

Hide

```
    elif user_data["entity"]=="Gastroenteritis":

        if "food" in user_data:
            if user_data["food"]==1:
                user_data["food"]=text
        #Process redflag###############
            redflag=[2,4,5,6,0]
            redflagage=['0-3mths', '3-6mths']
            redflaganswers=["3-6",">6",">3"]
            if user_data["SequenceCount"] in redflag :
                if text=="No" and user_data["SequenceCount"]==0:
                    if user_data["helpfulcount"]==2:
                        number=user_data["number"]
                        cd.recordred2(number,user_data)
                    else:
                        #print(user_data["helpfulcount"])
                        cd.recordend(user_data)
                    update.message.reply_text("Please come to our Childrens' Emergency at 5 Lowe
r Kent Ridge Road, 119074. One of our Specialist Nurse will be calling your registered Phone Num
ber to check on you in 10 mins.")
                    user_data.clear()
                    return ConversationHandler.END
                elif text=="Yes" and user_data["SequenceCount"]==0:
                    user_data["drink"]=text
                    update.message.reply_text("Advice for drink water\n")
                elif text=="Yes" and user_data["SequenceCount"]!=0:
                    update.message.reply_text("Please come to our Childrens' Emergency at 5 Lowe
r Kent Ridge Road, 119074. One of our Specialist Nurse will be calling your registered Phone Num
ber to check on you in 10 mins.")
  ##################################### Processing redflag
```

Producing relevant sequence of question: Initially, when prompted with open text, the chatbot will process it with ProcessConditionStart and subsequently, for all the remaining questions, ProcessCondition1 will loop back to itself until all the relevant questions are answered

Hide

```
            PROCESSCONDITIONSTART: [RegexHandler('^(Yes|No)$', ProcessCondition1, pass_user_data
=True),
                                    RegexHandler('^(0|1|2|>3)$', ProcessCondition1, pass_user_da
ta=True),
                                    RegexHandler('^(0-3|3-6|>6)$', ProcessCondition1, pass_user_
data=True),
                                    RegexHandler('^(0-3|>3)$', ProcessCondition1, pass_user_data
=True),
                                    MessageHandler(Filters.text,ProcessConditionStart, pass_user
_data=True)],
            PROCESSCONDITION1: [RegexHandler('^(Yes|No)$', ProcessCondition1, pass_user_data=Tru
e),
                                RegexHandler('^(0|1|2|>3)$', ProcessCondition1, pass_user_data=T
rue),
                                RegexHandler('^(0-3|3-6|>6)$', ProcessCondition1, pass_user_data
=True),
                                RegexHandler('^(0-3|>3)$', ProcessCondition1, pass_user_data=Tru
e),
                                RegexHandler('^(Helpful|Not Helpful)$', Helpful, pass_user_data=
True),
                                MessageHandler(Filters.text,ProcessCondition1, pass_user_data=Tr
ue)],
```

NewConditionQuestions.py file These are where the processing functions for relevant question generation is stored. Each question is numbered with an index and when running through the function, user_data[SequenceCount] is updated and question will be generated at the same time. Depending on SequenceCount number, the relevant question will be output. Note that because the updated SequenceCount and questions are output by the same function, when conditioning on question with index 1 in NewConditionQuestion file, the index for the same question in telegram_chatbot is index 2. Note that if the last index is 6, then the one alluded in the telegram_chatbot file is 0.

Hide

```
def ConstipationQuestion(value,SequenceCount,start): #value is word, SquenceCount is recording t
he question state, start is to let the function know if it is the first question being asked, be
cause if it is the sequence count will be updated to its rightful value.
    Question_Index={"not passing motion":0,"straining":1,"blood poo":2,"large stool":3,"painful"
:4,"bloated":5}
    Question_Dictionary={"0":"How many times is your child passing motion in 1 week?","1":"Does
 your child have to strain when passing motion?","2":"Is there any blood when your child passes
 motion?","3":"Does your child usually pass large sized stools?","4":"Is it painful when your ch
ild passes motion?","5":"Is your child's belly bloated?"}
    if start==1:
        #output value question and updated sequence count, when sequence count is 0, update to v
alue's index
        try:

             question_index=str(Question_Index[value]) #derive from current value
             SequenceCount=Question_Index[value] #update the sequence count to its rightful numb
er
             question=Question_Dictionary[question_index] #get question
        except KeyError:
             question_index=str(0) #derive from current value
             SequenceCount=0#update the sequence count to its rightful number
             question=Question_Dictionary[question_index] #get question
        if SequenceCount<5:
            SequenceCount+=1 #update sequence count to output
        else:
            SequenceCount=0
        return question,SequenceCount
    else:
        question_index=str(SequenceCount)
        question=Question_Dictionary[question_index]
        if SequenceCount<5:
            SequenceCount+=1
        else:
            SequenceCount=0
```

Training of Chatbot For details on the training of chatbot visit https://nlu.rasa.com/python.html (https://nlu.rasa.com/python.html) When the data is trained, in the json format, the model will appear under projects default. The training file appears in the same folder as RasaNLU, named demo-json1

Recording of conversation The recoding of conversation is under the NewConditionQuestion.py code. For each type of ending in the conversation to be recorded there is a different code. However, the differences are mainly in the ending sentences to be output to user. The main workings of the code are just conditioning on if the key exist in user_data and if it does, to record it into the pdf file.

Hide

```
def recordend(user_data):
    #record in file
        #generate random record number
    pdf = FPDF()
    pdf.add_page()
    pdf.set_xy(0, 0)
    pdf.set_font('arial', 'B', 13.0)
    number=random.randrange(0,10000000)
    filename="Patient "+str(number)+".txt"
    Record  = open(filename, "w")
    pdf.cell(60, 10, "Patient "+str(number), 0,1,)
    Record.write("Start time: "+user_data["starttime"]+"\n")
    pdf.cell(60, 10, "Start time: "+user_data["starttime"], 0,1,)
    Record.write("Age: "+user_data["age"]+"\n")
    pdf.cell(60, 10, "Age: "+user_data["age"], 0,1,)
    Record.write("Did child underwent surgery?: "+user_data["Surgery?"]+"\n")
    pdf.cell(60, 10, "Did child underwent surgery?: "+user_data["Surgery?"], 0,1,)
    if "surgery" in user_data.keys():
        Record.write("Type of Surgery: "+user_data["surgery"]+"\n")
        pdf.cell(60, 10, "Type of Surgery: "+user_data["surgery"], 0,1,)
    if "score" in user_data.keys():
        Record.write("FLACC score: "+str(user_data["score"])+"\n")
        pdf.cell(60, 10, "FLACC score: "+str(user_data["score"]), 0,1,)
    if "bleed" in user_data.keys():
        Record.write("Bleeding in wound?: "+user_data["bleed"]+"\n")
        pdf.cell(60, 10, "Bleeding in wound?: "+user_data["bleed"], 0,1,)
    if "puswound" in user_data.keys():
        Record.write("Pus in wound?: "+user_data["puswound"]+"\n")
        pdf.cell(60, 10, "Pus in wound?: "+user_data["puswound"], 0,1,)
    if "painmed" in user_data.keys():
        Record.write("Pain medicine working?: "+user_data["painmed"]+"\n")
        pdf.cell(60, 10, "Pain medicine working?: "+user_data["painmed"], 0,1,)
    if "drain" in user_data.keys():
        Record.write("Draining function working?: "+user_data["drain"]+"\n")
        pdf.cell(60, 10, "Draining function working?: "+user_data["drain"], 0,1,)
    if "string" in user_data.keys():
        Record.write("Is the string intact?: "+user_data["string"]+"\n")
        pdf.cell(60, 10, "Is the string intact?: "+user_data["string"], 0,1,)
        ....
    ts=time.time()
    endtime = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
    Record.write("End time: "+str(endtime)+"\n")
    pdf.cell(60, 10, "End time: "+str(endtime), 0,1,)
    Record.write("Conclusion:Patient was urged to go A&E")
    pdf.cell(60, 10, "Conclusion:Patient was urged to go A&E", 0,1,)
    pdf.output("Patient "+str(number)+".pdf", 'F')
    Record.close()
```