Navajyoti, International Journal of Multi-Disciplinary Research
Volume 5, Issue 1, August 2020

CHATBOT FOR WEATHER INFORMATION USING RASA

Sweety Lenka[1], Mary Harshitha[2], Swarnamugi M[3]

1Student, Department of MCA, Jyoti Nivas College, Bangalore, India

2 Student, Department of MCA, Jyoti Nivas College, Bangalore, India

3 Faculty, Department of MCA, Jyoti Nivas College, Bangalore, India

Abstract

The Chabot or a contextual Assistant is a computer program for human conversation simulation through audio or text messages. Apple's Siri, Microsoft's Cortana, Google Assistant, and Amazon's Alexa are few popular examples today. The chatbot is basically designed to provide the relevant output to the user, when an input is given like, responding to the user when questions are asked and also the most concentrated area of contextual assistant is that simulating the conversion in a realistic way. To simulate a realistic conversion, chatbot must understand the intention of the message and answer accordingly. To add on to the realistic conversion chatbot could be integrated with sentiment analysis.

In our research we have built a Weather Chatbot using RASA Core and RASA NLU that interacts with the user and also provides weather information. Here we concentrate on the **"context".** Context is the nothing but the way in which words are set or the way in which an event is set. Our Chatbot considers the context of the conversation thus it can gracefully handle unexpected dialogue turns and drive the conversation when the user drifts from the regular conversation path and improves over time by learning from what has been said during the conversation.

Keywords: Chabot, RASA, Neural Network, Long short term memory, Natural Language Processing, NLTK

I. Introduction

Contextual AI Assistants go far beyond Simple Notification Assistants and FAQ Assistants. Contextual AI Assistants consider the context of what has been said before which is a vital part of every natural conversation. If you look at how humans talk to each other. The context

matters. The chatbot should consider the context – "What the user said before, when and where and how they said should influence how the conversation goes. The chatbot should be able to handle unexpected user inputs and drive the conversations when the user drifts from the actual conversation path. The chabot should be capable enough to understand and respond to unexpected user queries.

Level 1: Notification Assistants

Notification Assistants - It is as simple as receiving notifications on your phone. This is how push notifications work on iOS and Android devices, with basic settings.

Level 2: FAQ Assistants

The chatbot lets the user ask simple queries, which is slightly different from regular FAQ pages. Few FAQ assistants can support multi-step conversation. Contextual AI Assistants are a big step further from FAQ's and Notifications. When humans talk to each other the content and also the context matters – What the user said before, when and where and how they said it should influence how the conversation goes. Understanding and responding to different user inputs is also referred to as context. This is how they differ from FAQ's and Notifications. They consider the context of what has been said before. The chatbots handle unexpected dialogue turns and drive the conversations when the user drifts from the regular conversation path and improve over time.

Rasa is a set of open source machine learning framework for building chat and voice contextual assistants. It consists of two components:

NLU: It is like the ear of the assistant. It understands what the user says. It understands what the user says. It takes input in the form of natural unstructured human language and extracts structured data in the form of intents and entities.

Intents are labels that are attached to user inputs based on the overall goals of the user's message.

For instance: User input Hello may have an intent greet.

Entities are piece of information that an assistant may need in a certain context.

For instance: My **name** is Sweety.

"name" is an entity so that user could be addressed throughout the conversation.

The classification of intent is done by **NLU** and extracts the entity from the user input and helps bot to understand what the user is saying.

Dialogue Management Component: It is also called as the core of the Contextual AI Assistant i.e. the brain. It takes in the input from the NLU and predicts the next best move using a model like LSTM neural network. Few keywords are used repeatedly in the implementation of the Contextual AI Assistants:

Intent — Intent is nothing but what the user is aiming for.

For example — if the user says "Reserve a table at Empire" the intent can be classified as reserving or to book the table.

Entity — Entity is to extract the vital information from the input.

From the example "Reserve a table at Empire" the entities extracted would be place and time. Place - Empire and Time - tonight.

Stories - Stories define the interaction between the user and chatbot in terms of intent and the action or the move taken by the chatbot. Like in the example above chatbot got the intent of reserving the table and entities like place and time but there is an entity missing - number of individuals and that would make the next best action by the chatbot.

Actions - Actions are the operations performed by the chatbot either asking details to get all the entities or integrating with some APIs or querying the database to get or save some data.

Rasa Natural Language Understanding (NLU) the bot should be first taught to understand the messages. For that, train the NLU model with inputs in a text format and extract structured data. This is achieved by defining the intents and providing a few other options where users might express them.

To make this work, define some files:

NLU training file: It contains training data in the form of inputs along with the mapping of intents and entities in each of them. The more varying examples provided, the better the bot's NLU capabilities become.

Stories file: It contains sample interactions. Rasa (Core) creates a model of interaction from

every story.

Domain file: This file the list of all the intents, entities, actions, templates. The templates are nothing but the sample chatbot reply which can be used as actions. In our proposed system, we are building a Contextual AI Assistant using Rasa from just an idea all the way to production. A Contextual AI Assistant's go far beyond simple FAQ interactions. We have built a Contextual AI Assistant which interacts with the user and also provides weather information. As they consider the context of what has been said before they can beautifully handle unexpected dialogue turns and drive the conversation when the user drifts from the regular conversation path and improve over time. Contextual AI Assistants are a big step further from FAQ's and Notifications.

II. Related Work

The existing chatbots used a sequence-to-sequence model [1] with attention mechanism that allows decoder more direct access to hidden state output by the encoder. For the RNNs,[2][3] the stacked LSTM cells of 2 layers are used. The encoder is the comment by the human, and the decoder is the output. It is assumed that in usual conversations, people listen to the first part and somewhat zone out to find the solution, the encoder is reversed so that the model can retain more information from the beginning of the remark.

Encoder takes in raw input text data. The output of the encoder becomes the input for the decoder. The model [4] uses a start token or an end token for encoders as well as decoders. The model produces the outputs by using the most likely token at every decoder procedure.

Training:[5] attention used in the decoding layers reduces the loss of the model by about 20% and increases the training time by about 20%. The model [6] performs outstandingly when the attention states are set with 0 (zeroes).The system is built on an LSTM classifier that has been trained on a huge data of questions and answers carefully prepared by the domain experts. The linguistic training bias enters into the human created training data due to particular phrases being used, with very few or zero variation, which biases the deep-learning classifier towards incorrect characteristics. Often the FAQs as confronted by the trainers are incomplete, and transferring linguistic variations across input output pairs can discover new input classes for which outputs are missing.

In another paper it is demonstrated that an variational auto-encoder [7] can be used to automatically generate linguistically novel inputs, which, (i) rectifies classifier bias when added to the training data set,(ii) discovers incompleteness in the set of solutions and (iii) improves the authenticity and accuracy of the base LSTM classifier, enabling it to learn from simpler training data.

III. PROPOSED WORK

In this paper, we are building a Contextual AI Assistant using Rasa. A Contextual AI Assistant's go far beyond simple FAQ interactions. They consider the context of what has been said before which is a vital part of each step in a natural conversation. We have built a Contextual AI Assistant that interacts with the user and also provides weather information. As they consider the context of what has been said before they can gracefully handle unexpected dialogue turns and drive the conversation when the user drifts from the regular conversation path and improve over time. Contextual AI Assistants are a big step further from FAQ's and Notifications.

IV. Methodology

LSTM Algorithm

The vital and most important component of the model is a recurrent neural network, which maps from dialog history directly to a distribution over system actions. The LSTM infers a representation of dialog history, which relieves the system developer from most of the manual feature engineering. The use of LSTM is to take actions in the real world on behalf of the agent who uses the system, it is optimized using supervised learning methodology, where a domain expert provides example dialogs which the LSTM should be trained with to imitate; or using reinforcement learning, where the system improves by interacting directly with the environment

Core Concept

- The core concept of LSTM is the cell state, and it's several other gates.
- The cell state transfers relevant data from the conversation. It is considered as the **"memory"** of the network.
- The cell state carries relevant data throughout communication. Thus data from the previous communication makes its way to the future communication .Thus our chatbot provides relevant information even when the user drifts from regular path.
- Thus as communication goes information gets added on making the chatbot give relevant responses making it like a human conversation.
- The gates learn what data is important to keep or forget during training avoiding compilation of huge data that are not relevant.

V. IMPLEMENTATION

When creating a conversational AI Assistant with RASA everything starts with training data.

Language Understanding

- We create some training data here, grouping user messages by their intents. The intent describes what the messages *mean*.
- Entities are labeled with square brackets and tagged with their type in parentheses. Syntax: [entity value](entity type):
- The Data directory contains two files:
- **nlu.md** - the file containing NLU model training examples. Includes intents, which are user goals, and example remarks represents these intents. The training data labels the entities, or particular vital keywords, the assistant extracts from the remarks.
- **stories.md** - the file containing story data.

```
## happy path                <!-- name of the story - just for debugging -->
* greet
  - utter_greet
* mood_great                <!-- user utterance, in format intent[entities] -->
  - utter_happy
* mood_affirm
  - utter_happy
* mood_affirm
  - utter_goodbye

## sad path 1                <!-- this is already the start of the next story -->
* greet
  - utter_greet             <!-- action the bot should execute -->
* mood_unhappy
  - action_retrieve_image
  - utter_cheer_up
  - utter_did_that_help
* mood_affirm
  - utter_happy

## sad path 2
* greet
  - utter_greet
* mood_unhappy
  - action_retrieve_image
  - utter_cheer_up
  - utter_did_that_help
* mood_deny
  - utter_goodbye

## strange user
* mood_affirm
  - utter_happy
* mood_affirm
  - utter_unclear
```

- **To properly train our NLU model:**

To enable our assistant to understand the intents and extract the entities from the NLU data file we need to build a model. This is done by defining a processing pipeline.
A pipeline is a sequence of processing steps which allows the model to learn the training data's underlying patterns. There are two pipelines available. Both pipelines are capable of performing intent classification and entity extraction.

```
language: "en"

pipeline: "pretrained_embeddings_spacy"
```

Word embedding - converts words to vectors. Like words are represented by like vectors, where the meaning is captured via this technique.

Training pipeline components use word embedding's to make text data understandable to the ML model.

Pretrained_embeddings_spacy - Uses**Pretrained_embeddings_spacy** - Uses the spaCy library to load pre-trained language models, which are used to represent every word in the user's question as word embeddings.
Advantages

- Increases the accuracy of the models, even if the training data provided is very little.

Supervised embeddings - The supervised embeddings pipeline trains the model from the very first stage using the information provided in the training data file.
Advantages:

- Could adapt to domain-specific dialogs, because the model is trained on the training data.
- Language-agnostic - Allows to build assistants in any language.
- Supports dialogs along multiple intents.

Pipelines

```
- name: "SpacyNLP"
- name: "SpacyTokenizer"
- name: "SpacyFeaturizer"
- name: "RegexFeaturizer"
- name: "CRFEntityExtractor"
- name: "EntitySynonymMapper"
- name: "SklearnIntentClassifier"
```


SpacyNLP loads the Language Model.
SpacyTokenizer takes the user input in an unstructured human language and splits them into tokens. A token could be a word.
SpacyFeaturizer is used for making the predictions.
CRFEntityExtractor It is used for entity extraction.
SklearnIntentClassifier is used for intent classification.

We train the model to recognize the intents, so that when the user feeds in data like "hello" to the bot, it will recognize this as a "greet" intent.

Use & evaluate the NLU model:

{'intent_evaluation':
{'predictions': [{'text': 'hey',
'intent': 'greet',
'predicted': 'greet',
'confidence': 0.9615294933319092},
{'text': 'hello',
'intent': 'greet',
'predicted': 'greet',
'confidence': 0.9637511968612671},
{'text': 'hi', 'intent': 'greet',
'predicted': 'greet',
'confidence': 0.9530026316642761},
{'text': 'good morning', '
intent': 'greet',
'predicted': 'greet',
'confidence': 0.9454197883605957},
{'text': 'good evening',

Adding dialogue capabilities

Writing Stories

A good place to start is by writing a few stories. These are example conversations that Rasa Core will learn from. The format works like this: A story starts with ## and you can give it a name. lines that start with * are messages sent by the user. Although you don't write the actual message, but rather the intent (and the entities) that represent what the user means.

Defining a Domain

The domain specifies the universe that your bot lives in. Here we list all of the intents and actions that pops up in the stories. This is also the place to write templates, which contain the messages your bot can send back.

Adding Custom API methods

We not only want to send back messages to the user, but also call the OpenWeatherMap API to get the current weather information of any location. Thus we have created custom actions that will be called once the bots ML model predicts them.

Training your Dialogue Model

Now comes the fun part! We're going to show Rasa Core the stories we wrote above, and train a model on these examples. In this case, the system is a neural network implemented in Keras i.e.(LSTM) which learns to predict which action to take next.

Starting up the bot

Now that we've trained the dialogue **and** language understanding models and saved them, we can start up an Agent which will handle conversations for us.

Talking to the Bot

We can start talking to the bot in natural language.

VI. TEST ANALYSIS

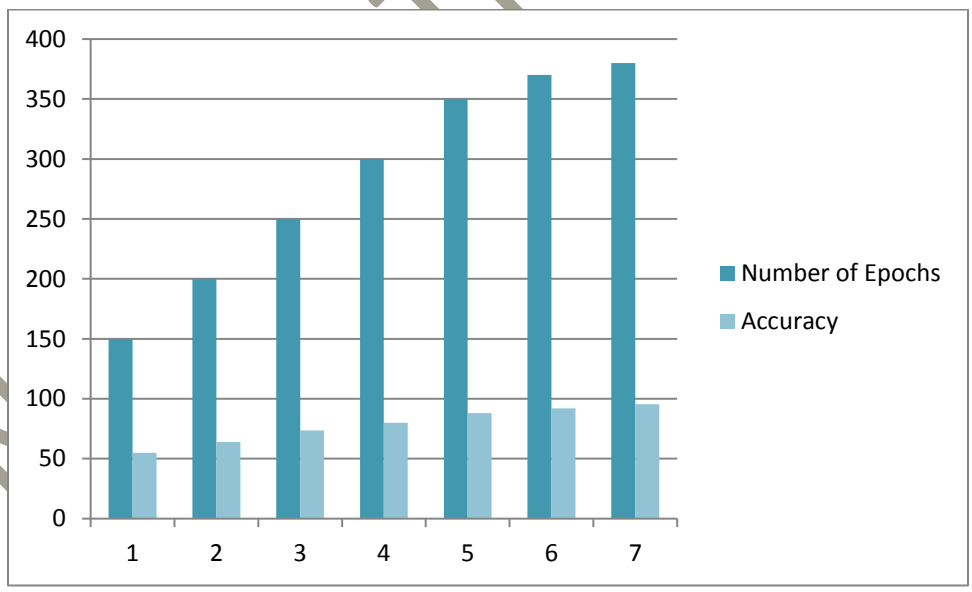
```
In [14]: from rasa_core.policies import KerasPolicy, MemoizationPolicy
         from rasa_core.agent import Agent

         agent = Agent('domain.yml', policies=[MemoizationPolicy(), KerasPolicy(validation_split=0.0, epochs=382)])

         # loading our neatly defined training dialogues
         training_data = agent.load_data('stories.md')

         agent.train(training_data)

         agent.persist('models/dialogue')

INFO:apscheduler.scheduler:Scheduler started
Processed Story Blocks: 100%|██████████| 8/8 [00:00<00:00, 219.11it/s, # trackers=1]
Processed Story Blocks: 100%|██████████| 8/8 [00:00<00:00, 123.99it/s, # trackers=6]
Processed Story Blocks: 100%|██████████| 8/8 [00:00<00:00, 105.37it/s, # trackers=10]
Processed Story Blocks: 100%|██████████| 8/8 [00:00<00:00, 97.41it/s, # trackers=11]
Processed actions: 301t [00:00, 268.95it/s, # examples=28]
INFO:rasa_core.policies.keras_policy:Fitting model with 171 total samples and a validation split of 0.0
```

```
Epoch 371/382
171/171 [==============================] - 0s 358us/sample - loss: 0.1042 - acc: 0.9298
Epoch 372/382
171/171 [==============================] - 0s 372us/sample - loss: 0.1179 - acc: 0.9240
Epoch 373/382
171/171 [==============================] - 0s 376us/sample - loss: 0.1178 - acc: 0.9298
Epoch 374/382
171/171 [==============================] - 0s 349us/sample - loss: 0.1117 - acc: 0.9181
Epoch 375/382
171/171 [==============================] - 0s 385us/sample - loss: 0.1143 - acc: 0.9298
Epoch 376/382
171/171 [==============================] - 0s 398us/sample - loss: 0.1168 - acc: 0.9006
Epoch 377/382
171/171 [==============================] - 0s 504us/sample - loss: 0.1146 - acc: 0.9123
Epoch 378/382
171/171 [==============================] - 0s 477us/sample - loss: 0.1130 - acc: 0.9240
Epoch 379/382
171/171 [==============================] - 0s 467us/sample - loss: 0.1117 - acc: 0.9240
Epoch 380/382
171/171 [==============================] - 0s 475us/sample - loss: 0.1215 - acc: 0.9006
Epoch 381/382
171/171 [==============================] - 0s 483us/sample - loss: 0.1137 - acc: 0.9064
Epoch 382/382
171/171 [==============================] - 0s 421us/sample - loss: 0.1012 - acc: 0.9532
```

```
Your bot is ready to talk! Type your messages here or send 'stop'

hello bot
Hi. My name is Pro Bot. How can I help you today?

doing great?
Great carry on!

actually I'm sad
Here is something to cheer you up:the weather was sunny and also my friend name is sunny

Did that help you?

yeah kind of thanks bye
Bye
```

```
Your bot is ready to talk! Type your messages here or send 'stop'

hi bot
Hi. My name is Pro Bot. How can I help you today?

can you give me weather today
To find the nearest None I need your address.

Bangalore
<pyowm.weatherapi25.weather.Weather - reference time=2019-11-20 05:48:44+00, status=clouds, detailed status=scattered clouds>

Wind speed( Bangalore ):  {'speed': 4.1, 'deg': 70}

Temperature( Bangalore ):  {'temp': 25.95, 'temp_max': 27.22, 'temp_min': 25.0, 'temp_kf': None}
```

GRAPH REPRESNTING THE ACCURACY OF THE WEATHER CHATBOT IMPLEMENTED USING LSTM



VII. Conclusion

In our proposed system, we are building a Contextual AI Assistant using Rasa from just an idea. They consider the context of what has been said before which is a very important part of every natural conversation. We would like to improve the chatbot that would consider the time context.

References

1. https://medium.com/@itsromiljain/build-a-conversational-chatbot-with-rasa-stack-and-python-rasa-nlu-b79dfbe59491
2. A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks, 2018 IEEE 15th Internatio nal Conference on e-Business Engineering (ICEBE)
3. Ayesha Shaikh1, Geetanjali Phalke2, Pranita Patil3, Sangita Bhosale4, Jyoti Raghatwan5 "A Survey On Chatbot Conversational Systems", IJESC
4. Mayur Patidar, Puneet Agarwal, Lovekesh Vig, and Gautam Shro. "Correcting Linguistic Training Bias in an FAQ-bot using LSTM-VAE"
5. Panitan Muangkammuen ; Narong Intiruk ; Kanda Runapongsa Saikaew. "Automated Thai-FAQ Chatbot using RNN-LSTM". Publisher: IEEE
6. Ankil Shah ; Bhargav Jain ; Bhavin Agrawal ; SaurabhJain ; Simon Shim "Problem solving chatbot for data structures," Publisher: IEEE,
7. Dongkeon Lee ; Kyo-Joong Oh ; Ho-Jin Choi "The chatbot feels you - a counseling service using emotional response generation." Publisher: IEEE,
8. Andreas Lommatzsch and Jonas Katins TU Berlin, DAI-Labor, Ernst-Reuter-Platz. "An Information Retrieval-based Approach for Building Intuitive Chatbots for Large Knowledge Bases"
9. Heriberto Cuáyahuitl ; Donghyeon Lee ; Seonghan Ryu ; , **"**Deep Reinforcement Learning for Chatbots Using Clustered Actions and Human-Likeness Rewards, "Publisher: IEEE.
10. Sarthak V. Doshi1 , Suprabha B. Pawar2 , Akshay G. Shelar3 , Shraddha S. Kulkarni4, " Artificial Intelligence Chatbot in Android System using Open Source Program", Publisher: International Journal of Advanced Research in Computer and Communication Engineering
11. Chih-Wei Lee, Yau-Shian Wang, Tsung-Yuan Hsu, Kuan-Yu Chen, Hung-Yi Lee, Lin-Shan Lee. "Scalable Sentiment for Sequence-To-Sequence Chatbot Response With Performance Analysis"
12. https://blog.rasa.com/