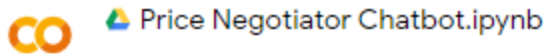

Prototyping

Model architecture.

The price negotiator project has two components, one is the eCommerce platform that works on Django, a website framework based on Python language, the second section of our project is an AI chatbot that connects with the same database, that e-commerce website uses, to get the discounts range and build the conversion with clients. The chatbot uses a specific AI algorithm to process the client message and response.

For deploying purposes, we have taken a copy of the code that was used to develop the AI part of the chatbot and implemented it on Colaboratory, or "Colab" .



colab file link:

https://colab.research.google.com/drive/1j_NP-b7WJ5g5ligVwJkJKt-Yd7istTa7?usp=sharing

Our AI model is a Retrieval-Based chatbot. This chatbot uses some heuristics to select an appropriate response from a library of predefined responses. This architectural model of a chatbot is easier to build and much more reliable. Though there cannot be 100% accuracy of responses, you can know the possible types of responses and ensure that no inappropriate or incorrect response is delivered by the chatbot. This bot considers the message and context of the conversation to deliver the best response from a predefined list of messages. To implement the chatbot, we used Keras, which is a Deep Learning library, NLTK, which is a Natural Language Processing toolkit, and some other helpful libraries.

Feeding data.

Data format :JSON (JavaScript Object Notation)

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": [
        "Hi there",
        "How are you",
        "Is anyone there?",
        "Hey",
        "Hola",
```

```
    "Hello"
  ],
  "responses": [
    "Hello",
    "Good to see you again",
    "Hi there"
  ],
  "context": [
    ""
  ]
},
{
  "tag": "goodbye",
  "patterns": [
    "Bye",
    "See you later",
    "Goodbye",
    "Nice chatting to you, bye",
    "Till next time"
  ],
  "responses": [
    "See you!",
    "Have a nice day",
    "Bye! Come back again soon."
  ],
  "context": [
    ""
  ]
},
{
  "tag": "thanks",
  "patterns": [
    "Thanks",
    "Thank you",
    "That's helpful",
    "Awesome, thanks",
    "Thanks for helping me"
  ],
  "responses": [
    "Happy to help!",
```

```
        "Any time!",
        "My pleasure"
    ],
    "context": [
        ""
    ]
},
{
    "tag": "noanswer",
    "patterns": [],
    "responses": [
        "Sorry, can't understand you",
        "Please give me more info",
        "Not sure I understand"
    ],
    "context": [
        ""
    ]
}
]
```

Model Testing.


User Input	Expected output	Actual Output
Hello	"Hello", "Good to see you again", "Hi there"	Hi there
Till next time	"Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"	See you later
dkmvndknkj	"Sorry, can't understand you", "Please give me more info", "Not sure I understand"	Sorry, can't understand you

Colab interface(the AI model implemintation) :

```
CHAT START (BLANK AND ENTER TO STOP)
YOU:hi
CHATBOT: Hello
YOU:thanks
CHATBOT: Happy to help!
YOU:meme
CHATBOT: Please give me more info




CHAT END
```

Django Project interface (chatbot AI model embedded in the ecommerce website)

 Price Negotiator

Price Negotiation session with Mervat Mustafa

Your Cart

Sno.	Image	Product	Price	Quantity	Total
1		Baby Frock	80 CAD	1	80 CAD
2		Baby Boy Dress	45 CAD	1	45 CAD
3		Bands	28 CAD	1	28 CAD

Let's chat? - AI robot

hi

Hello. It is our pleaser to offer you 2% discount , the tatol amount will be :149.94 , are you happy with this offer?

Yes

We are happy that we could help you today. You got 2% discount , the tatol amount is :149.94 Please complete the checkout process.

thanks

Happy to help!

Type a message...

Evaluate models and consider alternatives.

The current model works satisfactorily. The AI takes input from intents where the tag contains a list of patterns and responses. We tokenize each pattern and add the words in a list. Also, we create a list of classes and documents to add all the intents associated with patterns. The chatbot will capture the user message and perform some preprocessing. The model will then predict the tag of the user's message, and will randomly select the response from the list of responses in our intents file.

We train the model, by converting each input pattern into numbers. Using the WordNet lemmatizer from the nltk library we reduce words to its base form, where plays, played, playing become play. Then return a bag of words by creating a list of zeroes of the same length as the total number of words. We set value 1 to only those indexes that contain the word in the patterns. In the same way, we create the output by setting 1 to the class input the pattern belongs to. We use our word.pkl file, containing the vocabulary of our project and classes.pkl containing the total entities, to classify our data.

Next is the architecture of our model, which consists of a neural network with 3 layers. The first layer has 128 neurons, the second one has 64 and the last layer will have the same neurons as the number of classes. The dropout layers are introduced to reduce overfitting of the model. We have used the SGD optimizer and fit the data to start the training of the model. After the training of 200 epochs is completed, we then save the trained model using the Keras model.save("chatbot_model.h5") function.

In the end, the chatbot takes an input message, creates a bag of words array, filters it through sorting and returns the input for intents where the responses with respect to particular user input is given as output.

The cases of human error are a natural part of the process. Hence, we can have a more forgiving and understanding chatbot that has more functions for its users.

Comprehension capabilities:

One way is to increase the scope and functionality of the chatbot by spellcheck like Google has. For instance, the chatbot outputs correct output even if the user inputs with incorrect spellings. Along with spell check from high quality data, chatbots may aim for two types of intent understanding. It can have 'text-based understanding' to quickly understand the questions and statements the user is texting. The chatbot can also have 'balanced text-use'. This means that a chatbot will use a combination of both short descriptions and engaging content, like rich media, to hold the user's attention.

Speed: A chatbot can be integrated with a knowledge-based database and programmed to fetch information and respond quickly. Measuring the response rate of the chatbot plays an important role when it comes to speed.

Interoperability: A chatbot can be designed in a way that different ecommerce platforms can make use of it. The aim can be to make the chatbot like a readymade software tool so that small level businesses can plug them into their systems with minimum code requirements.

Scalability: A chatbot can also be designed to be scalable. As data increases, small things become hard to implement. After the initial stages of the chatbot, the next stage will be to plan for demands with respect to users in millions.